Reliable and Energy Efficient Resource Provisioning in Cloud Computing Systems

Yogesh Sharma

A dissertation submitted in total fulfilment of the requirements for the degree of

Doctor of Philosophy

School of Computing, Engineering and Mathematics
Western Sydney University

August 2018

Reliable and Energy Efficient Resource Provisioning in Cloud Computing Systems

Abstract

Reliable and Energy Efficient Resource Provisioning in Cloud Computing Systems

by

Yogesh Sharma

Doctor of Philosophy in Information and Communication Technology

Western Sydney University

Bahman Javadi, Principle Supervisor

Weisheng Si, Daniel Sun , Co-supervisors

Cloud Computing has revolutionized the Information Technology sector by giving computing a perspective of service. The services of cloud computing can be accessed by users not knowing about the underlying system with easy-to-use portals. To provide such an abstract view, cloud computing systems have to perform many complex operations besides managing a large underlying infrastructure. Such complex operations confront service providers with many challenges such as security, sustainability, reliability, energy consumption and resource management. Among all the challenges, reliability and energy consumption are two key challenges focused on in this thesis because of their conflicting nature. Current solutions either focused on reliability techniques or energy efficiency methods. But it has been observed that mechanisms providing reliability in cloud computing systems can deteriorate the energy consumption. Adding backup resources and running replicated systems provide strong fault tolerance but also increase energy consumption. Reducing energy consumption by running resources on low power scaling levels or by reducing the number of active but idle sitting resources such as backup resources reduces the system reliability. This creates a critical trade-off between these two metrics that are investigated in this thesis.

To address this problem, this thesis presents novel resource management policies which target

the provisioning of best resources in terms of reliability and energy efficiency and allocate them to suitable virtual machines. A mathematical framework showing interplay between reliability and energy consumption is also proposed in this thesis. A formal method to calculate the finishing time of tasks running in a cloud computing environment impacted with independent and correlated failures is also provided. The proposed policies adopted various fault tolerance mechanisms while satisfying the constraints such as task deadlines and utility values. This thesis also provides a novel failure-aware VM consolidation method, which takes the failure characteristics of resources into consideration before performing VM consolidation. All the proposed resource management methods are evaluated by using real failure traces collected from various distributed computing sites. In order to perform the evaluation, a cloud computing framework, 'ReliableCloudSim' capable of simulating failure-prone cloud computing systems is developed. The key research findings and contributions of this thesis are:

1. If the emphasis is given only to energy optimization without considering reliability in a failure prone cloud computing environment, the results can be contrary to the intuitive expectations. Rather than reducing energy consumption, a system ends up consuming more energy due to the energy losses incurred because of failure overheads.

2. While performing VM consolidation in a failure prone cloud computing environment, a significant improvement in terms of energy efficiency and reliability can be achieved by considering failure characteristics of physical resources.

3. By considering correlated occurrence of failures during resource provisioning and VM allocation, the service downtime or interruption is reduced significantly by 34% in comparison to the environments with the assumption of independent occurrence of failures. Moreover, measured by our mathematical model, the ratio of reliability and energy consumption is improved by 14%.

Statement of Authentication

The work presented in this thesis is, to the best of my knowledge and belief, original except as acknowledged in the text. I hereby declare that I have not submitted this material, either in full or in part, for a degree at this or any other institution.

Yogesh Sharma

Acknowledgments

It is my pleasure to acknowledge the role of a number of wonderful individuals, who were instrumental in helping me complete my Ph.D. research. This thesis would have not been possible without their support.

First and foremost, I would like to express my sincere gratitude to my supervisor Dr. Bahman Javadi, who trusted me and awarded me an opportunity to work with him. Ever since, he supported me academically and emotionally through the rough road to finish this thesis. Secondly, I would like to express my appreciation to my co-supervisors Dr. Weisheng Si and Dr. Daniel Sun from Data61, CSIRO, Australia for being my major advisors. Their constructive feedback and critical comments have significantly enhanced the quality of my writing.

I would also like to express my respect to Dr. Rodrigo N. Calheiros for providing me technical advice and suggestions for the implementation part of this thesis. A special acknowledgement goes to my colleague and very good friend, Maria Mikhail. She is a true friend and an amazing person in too many ways. Being a non-specialist of the field, she used to provide her feedback after reading my works by highlighting the confusingly constructed sentences, which improved the clarity and quality of this thesis.

Thanks to Raed and Mohammad, my great office mates and team members who have been supportive in every way. Thank you also to Nabil and Guang for four years of technical computing support. I am grateful to Rosemary for advising me how to use Latex. My thanks also goes to administrative staff members in the School of Computing, Engineering and Mathematics (SCEM), especially Veena and Cheryl for their support and help.

I wish to acknowledge the Western Sydney University and Data61-CSIRO for granting the scholarships and travel supports that enabled me to conduct the research for this thesis, including attending international conferences.

I am immensely grateful to my school time teacher Bharti Sewak, Dr. Jyotsna Sengupta (Punjabi University, Patiala) and my music teacher Robin Gupta for their encouragement and inspiration to move forward and keep learning. I would like to express my sincere thanks to Huda Tanyous,

Sobhi Mikhail, Moura Tooma, David Di Lenno, Jujhvir Saini, Gagandeep Singh, Gurpinder Sidhu, Shivjeet Singh, Gurjot Kingra, Gurpreet Sarao, Nitin and Vandana Setia, Paramdeep Narain, Hetal and Barnie, who helped me through tough times and filled me with confidence and strength, when I needed it the most.

On a personal note, I would like to express my sincerest thanks to my maternal uncle Ramesh Dixit, who supported me financially in the beginning. Without his moral and financial support, I would not have been able to commence my Ph.D. To Kamal Dixit, thank you for always being a wonderful role model and to Gopal Dixit, I greatly appreciate your constant support and encouragement.

I would like to thank my mother and younger brother for their unfailing love, patience and support. They stood beside me and helped me in many ways while I conducted my research. Finally, I would like to thank my late father who supported my education from the very beginning. I will be forever grateful for his support. I wish he could see me achieve this amazing feat.

Yogesh Sharma
August 2018

Contents

List of Figures

List of Tables

Chapter 1

Introduction

CLOUD computing is a paradigm delivering computing as a service to end users and helping them to reduce IT related operational costs and complexities. Cloud service providers offer computing power using variety of service models such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). Users also have the option of different kind of deployment models such as private clouds, public clouds and hybrid clouds. Once the users switch to cloud, they get unlimited amount of computing and storage resources and billed according to their usage. Due to the elastic nature of cloud computing systems, users can dynamically provision and unprovision the computing resources according to their requirements with minimum human intervention. Because of such benefits, almost every sector such as social networking, defence, scientific computing, finance and medical systems are adopting cloud computing services at highest rate than ever. A recent report published by IDG communications found that 73% of organisations have at least one application running on the cloud and another 17% are planning to do so in next 12 months [72]. To make the services convenient and easily accessible, cloud computing services are provided as an abstract view of the system by using easy to use web portals such as AWS while hiding complex operations underneath. In order to provide such abstraction of the system, cloud service providers confront with multiple challenges such as operational expenses, reliability, energy efficiency, security and scalability.

Among the challenges, reliability is one of the major challenge that cloud computing industry is facing in these days. Reliability in this thesis is explored from the perspective of failures, although it can be explored from the perspective of security as well. Failures in complex computing systems such as clouds are inevitable. A failure or down-time in cloud computing services can cost huge to organizations in terms of both financial losses and reputational damages. In October 2013, Knight Capital′s cloud based automatic stock trading software went down for 45 minutes, which costed the company $440 million and the company had lost its 75% equity value [21]. To mitigate such failure losses and to make the cloud services available uninterruptedly, failure or fault tolerance plays a vital role.

The most common technique to provide fault tolerance in cloud computing systems currently in practice is the resource redundancy such that installation of backup resources or secondary resources. Whenever, a primary resource goes down because of a failure, secondary resource takes charge and begin to serve. However, adding extra resources increases the energy consumption more steeply than reliability and has a direct impact on the profit margins of the service providers and users. Energy consumption of cloud computing systems is already a major challenge. Approximately, 45% of total operational expenses of IBM data centers goes in electricity bills [13]. It has been reported that servers mounted in Microsoft's cloud based data centers consumes approximately 2 terawatt-hours (TWh) of energy per year for which the company pays approximately $2.5 billion per year as electricity bills [9]. Most of the servers in such data centers are sitting idle and are deployed to accommodate peak load in order to avoid an outage [158]. Cloud computing infrastructure is also a major contributor of carbon content to the environment and is expected to generate up to 5.5% of the world′s carbon emissions by 2025 [99].

The energy consumption can be reduced by decreasing the number of active but idle or under-utilized resources (backup resources), which on the contrary reduces the reliability of the systems. In case, if a failure will occur and no backup resource is available then all the running VMs and corresponding tasks will be recreated and restarted from the beginning. This will increase the processing overheads dramatically and which in return will have the huge impact on the service quality (QoS) of cloud computing systems. This creates a critical trade-off between the reliability and

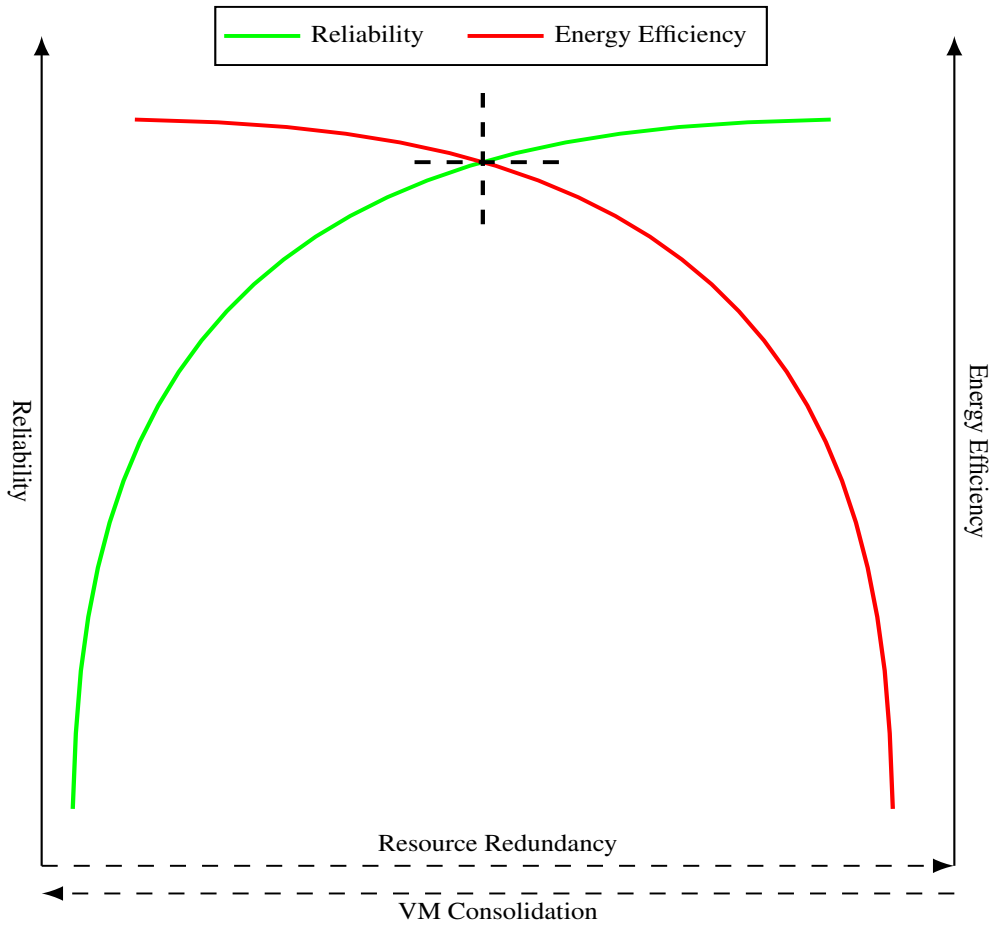


Figure 1.1: Reliability-Energy Trade-off. The figure shows the impact of manipulation of number of resources on the reliability and energy efficiency of cloud computing systems.[1]

energy efficiency of cloud computing systems (Figure 1.1) and needs a careful investigation, which is the motivation behind this thesis. The main objective of this thesis is to devise new resource provisioning and VM placement policies by using which the reliability and energy efficiency of failure prone cloud computing systems will be increased.

[1]Figure 1.1 is not obtained by any experimental study. It is simply a pictorial representation of trade-off between reliability and energy efficiency in cloud computing systems.

1.1 Contributions

In order to optimize the reliability and energy consumption in cloud computing systems, various mathematical models and resource management methods are proposed. Following are the key contributions of this thesis

1. **Survey and Taxonomy:** This thesis provides comprehensive taxonomies of existing techniques for reliability and energy efficiency and their trade-off in cloud computing systems. On the basis of the taxonomies, a survey and classifications of resource failures, fault tolerance mechanisms and energy management mechanisms in cloud computing is presented. The survey and taxonomy is intended to identify the future research and developments by highlighting the research gaps in trade-off between the reliability and energy efficiency in cloud computing systems.

2. **Statistical Analysis and Mathematical Models:** Statistical analysis of real failure traces gathered from Los Alamos National Lab (LANL) and Grid5000 distributed computing infrastructures is performed to study the patterns of availability and unavailability events. On the basis of the analysis, reliability of cloud computing systems is modeled and mathematical models are proposed to measure the impact of failures on the application finishing time and energy consumption of the system.

3. **Novel Resource Management Policies:** Resource and virtual machine (VM) management policies with integrated failure prediction and fault tolerance methods are proposed to increase the reliability and energy efficiency of failure prone cloud computing systems. The proposed policies are used to mitigate the impact of the occurrence of failures while ensuring the quality of services (QoS) and reducing the energy consumption of cloud computing infrastructure.

    a) Chapter 3 investigates how to provision the resources and allocate the VMs in order to reduce the energy consumption and increase the reliability in a failure prone cloud computing environment. To achieve the objective, three list based greedy policies

for resource provisioning and VM allocation integrated with checkpointing as fault tolerance are proposed. The policies takes the failure characteristics and power profiles of physical resources before performing resource provisioning and VM allocation.

b) Chapter 4 targets to reduce the number of provisioned resources in order to reduce the operational expenses dynamically while maximizing the reliability and minimizing the energy consumption of cloud computing systems. In response to the objective, a failure-aware VM consolidation mechanism is proposed, which takes the occurrence of failures and the hazard rate of physical resources into consideration before performing VM consolidation. Besides employing the policies proposed in chapter 3 for resource provisioning and VM allocation, a failure prediction technique based on exponential smoothing is proposed to trigger two fault tolerance mechanisms such as VM checkpointing and VM migration.

c) Chapter 5 is focused on improving the reliability and energy efficiency of cloud computing systems in the presence of correlated failures. To identify and manage the occurrence of correlated failures, a novel methodology is proposed using cluster analysis techniques. The results provided by cluster analysis are used to design correlated failure-aware energy efficient resource provisioning policies for failure prone cloud computing environment.

4. **Reliable and Energy Efficiency Cloud Computing Framework:** This thesis presents the design and architecture of a cloud computing framework using which the system can be deployed in a reliable and energy efficient manner.

1.2 Thesis Organization

This thesis includes seven chapters which are organized as shown in Figure 1.2. The overview of the organization is as follows

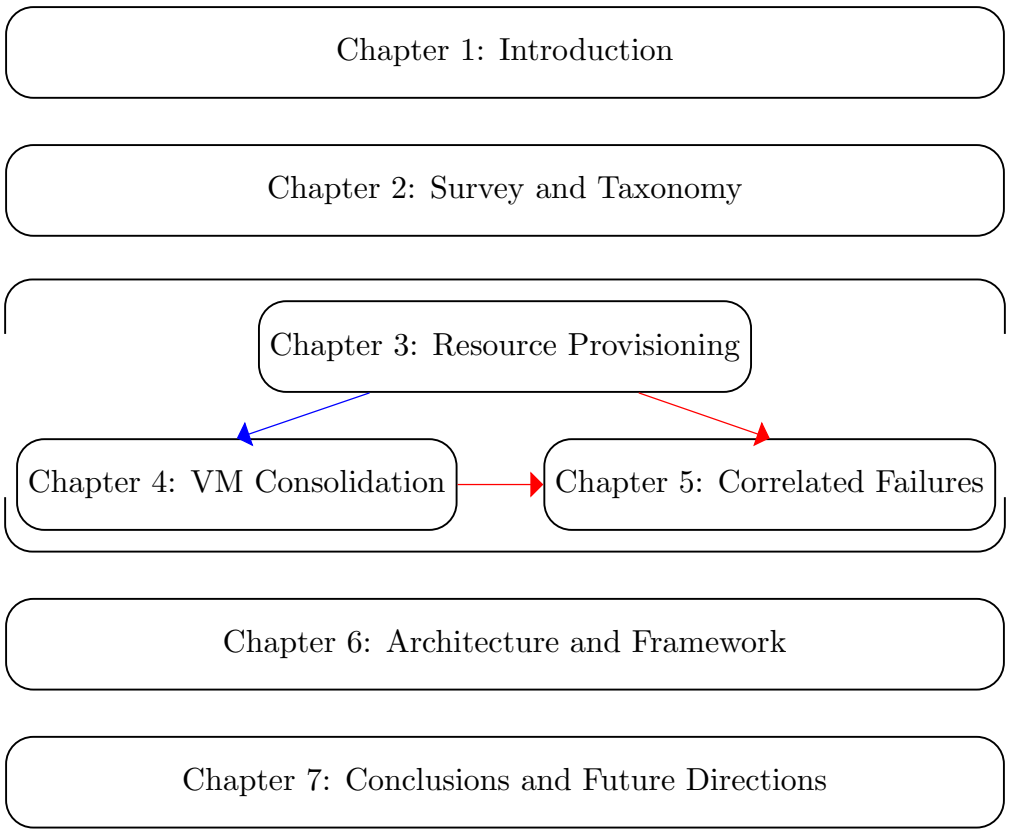


Figure 1.2: Thesis Organisation

- Chapter 2 presents a comprehensive survey and taxonomies covering various reasons for the occurrence of failures, fault tolerance and energy efficiency techniques and mechanisms for cloud computing systems. In the chapter, future research directions are also advised by identifying the research gaps in trade-off between the reliability and energy efficiency in cloud computing systems.

  – Sharma Y., Javadi B., Si W. and Sun D. "Reliability and energy efficiency in cloud computing systems: Survey and taxonomy". Journal of Network and Computer Applications 74(2016), pp. 66-85 [137].

- Chapter 3 provides mathematical models for both reliability and energy consumption of cloud computing systems and analyses their interplay besides a formal method to calculate the fin-

ishing time of tasks running in a failure prone cloud computing environment. Three resource provisioning and virtual machine (VM) allocation policies such as Reliability Aware Best Fit Decreasing (RABFD), Energy Aware Best Fit Decreasing (EABFD), Reliability-Energy Aware Best Fit Decreasing (REABFD) incorporating mathematical models are proposed while using checkpointing as fault tolerance mechanism.

  - Sharma Y., Javadi B., Si W. and Sun D. "Reliable and Energy Efficient Resource Provisioning and Allocation in Cloud Computing". Proceedings of the 10th IEEE/ACM International Conference on Utility and Cloud Computing (UCC'17), Austin, Texas, USA [138] (Best Paper Finalist).

- Chapter 4 proposes a failure-aware VM consolidation mechanism, which takes the failure characteristics of physical resources into consideration before performing VM consolidation. To provide fault tolerance both reactive and proactive mechanisms, i.e., VM migration and VM checkpointing, respectively are used which get triggered on the basis of the results provided by an exponential smoothing based failure prediction mechanism.

  - Sharma Y., Javadi B., Si W. and Sun D. "Failure-aware Energy-efficient VM Consolidation in Cloud Computing Systems". Future Generation Computer Systems *(Accepted on 27-11-2018)*.

- Chapter 5 proposes the mathematical models and resource management policies for regulating the reliability and energy consumption jointly under correlated failures in cloud computing systems. Statistical cluster analysis and time series analysis based techniques are employed to explore failure correlation and to predict the occurrence of correlated failures, respectively while using real failure traces.

  - Sharma Y., Javadi B., Si W. and Sun D. "Dynamic Resource Provisioning for Sustainable Cloud Computing Systems in the Presence of Correlated Failures". IEEE Transactions on Sustainable Computing *(under review)*.

- Chapter 6 presents the design and architecture of a cloud computing framework using which the system can be deployed in a reliable and energy efficient manner. The proposed architecture is realised by extending the 'CloudSim' simulator to simulate failure-prone cloud computing environment.
- Chapter 7 concludes and provides the directions for future work.

Chapter 2

Survey and Taxonomy

With the increasing trend of acceptance of cloud computing from educational institutions to business organizations, underlying computing infrastructure is also expanding with a great pace. Due to such expansion, cloud computing systems confront service providers and users with many challenges such as security, reliability, sustainability, energy and resource management. Among all the challenges, reliability and energy efficiency are two key conflicting challenges that need careful attention and investigation. The recent works are either focused on the reliability techniques or energy efficiency methods in cloud computing. This chapter presents a thorough review of existing techniques for reliability and energy efficiency and their trade-off in cloud computing. The classifications are also discussed and provides in-depth taxonomies on resource failures, fault tolerance mechanisms and energy management mechanisms in cloud systems. Moreover, various challenges and research gaps in trade-off between reliability and energy efficiency are identified for future research and developments.

2.1 Introduction

CLOUD computing is an ongoing revolution in information and communication technology (ICT) that uses virtualization technology to provide a powerful and flexible computing

environment. In a Gartner report published in January 2013, the growth of public cloud services will make it a $155 billion market and by the end of 2016, it is expected to grow to $210 billion [45]. Although cloud computing makes the computing reliable, dynamic, fast and easy, it is still facing numerous challenges because of its large-scale and complex architecture. Considering the scale and complexity of cloud data centres, reliability and energy efficiency are two key challenges that need careful attention and investigation. *Reliability of cloud computing systems can be defined in the context of security or in the context of resource and service failures.* Due to the complexity of the cloud architecture, failures are inevitable. It has been argued that a system with 100000 processors experience a failure every couple of minutes [50]. In cloud computing, failures could occur due to multiple reasons such as hardware failure, software failure, etc. (Figure 2.1). A failure in the services of a cloud costs significantly for both providers and customers. In a survey of 63 Data Centres done by Ponemon institute [73] in 2016, it has been reported that the average down-time cost of each data centre rose to $740,357 from $500,000 in 2010 (38% increase). Every hour, the business sector is expected to lose around $108,000 and according to the Information week, each year IT outages result in the revenue loss of more than $26.5 billion [69]. Provisioning of cloud resources accurately according to the demand of the applications plays a crucial role to make the cloud computing systems reliable and energy efficient. In cloud computing, it is hard to predict the requirement of resources accurately before or during submission of an application or task. Sometimes the provisioned resources remain underutilized or become over utilized. The average utilization of resources in cloud based data centers is less than 30% [105]. In case of underutilized resources, task or virtual machine (VM) consolidation is performed by migrating the running VMs to other physical resources in order to put the underutilized resources on sleep mode or to turn them off so as to reduce the energy consumption or other running costs [31]. In the case of over-utilization, the running tasks are migrated to other resources to keep the load of over-utilized resources below to a specific threshold to immunise them from failures or crashes.

However, the energy requirement to operate the cloud infrastructure is also increasing in proportion to the operational costs. Approximately, 45 percent of the total operational expenses of IBM data centres goes in electricity bills [132]. According to the Gartner, the electricity

consumption by cloud based data centres will increase to 1012.02 Billion kWh by 2020. In 2013, data centers alone in U.S. consumed 91 billion kilowatt-hours, which is enough to power all the households of New York City twice over and if this trend will continue then the consumption will reach 140 billion kWh by 2020, a 35% increase [39]. The energy that the U.S. based data centers are consuming is equal to the electricity produced by 34 power plants each of 500 megawatts capacity and if this can't be reduced then 17 new power plants will need to be established by 2020 to power the data centers [148]. The electricity or energy consumption in cloud infrastructures is very inefficient and there are several types of wastes at different levels such as infrastructure level or system level [115]. At the infrastructure level, half of the energy provided to a data centre is consumed by the cooling infrastructure and at the system level, 50% of the energy is consumed when systems are in idle state. These types of waste cause financial loss to both providers and users.

Cloud computing infrastructure is a major contributor to the carbon content of the environment as well. Along with many contributors of carbon emissions in the environment, the contribution of IT infrastructure is equal to the aviation industry. U.S. based data centers emit 100 million metric tonne of carbon content each year and will increase to 1034 metric tonne by 2020 [32]. As the energy consumption, heat release and carbon footprint from large computing infrastructures has increased, researchers are under great pressure to find new ways of decreasing energy consumption. In the last few decades, the primary focus of researchers and designers was on optimizing the performance of the system in terms of speed, space and efficiency. However, concerns about the energy consumption and carbon footprint intensified recently. In January 2015, Amazon has announced the construction of 150 MW wind farm which will produce approximately 500000 MWh of wind power [6]. The energy generated by the wind farm will be used to power the current and future cloud based AWS (Amazon Web Services) data centers. Microsoft had also made a carbon neutral commitment in 2012 by promising to achieve zero emission of carbon content by their data centers and software development labs. Google, IBM and other cloud vendors are also working to make the cloud services and cloud based data centers energy efficient and eco-friendly.

All the above facts and figures of failure and energy consumption lead to the requirement of

management of cloud resources in a fault-tolerant and energy-efficient way. In response to this, various researchers worldwide have proposed many architectures, algorithms and policies to make the cloud computing environment reliable and energy efficient. However, there is very limited research on the trade-off between reliability and energy efficiency in cloud computing systems (subsection 2.5.1). Considering both parameters at the same time would open new opportunities and challenges in the area of resource management in cloud systems. This chapter gives a comprehensive survey of the research done in the field of reliability and energy efficiency followed by an analysis of the trade-off between these two metrics in cloud computing systems.

2.2 Failures in Cloud and Distributed Computing Environments

In this section, the classification of failures in cloud and distributed computing systems is reviewed. The failure correlation as well as causes for failures are also discussed.

A Failure is defined as an event in which the system fails to operate according to its specifications. A system failure occurs, when a system deviates from fulfilling its normal system function for which it was aimed at.

According to Google [13], the cost for each repair of failure includes $100 for technician's time and 10% of the total cost of server ($200), which reaches to $300 per repair. Therefore, the cost of repairing the hardware exceeds its buying cost after only 7 repairs. Sound knowledge of the type of failure and causes of failure will help computer scientists and computer engineers to design more scalable algorithms and to deploy infrastructure in more fault tolerable way. This will help to reduce the repair/replacement cost and engineering expenditures and makes the computing, specifically service computing such as cloud computing, more reliable. Failures in cloud computing systems result in loss of business due to the diversion of users to other vendors.

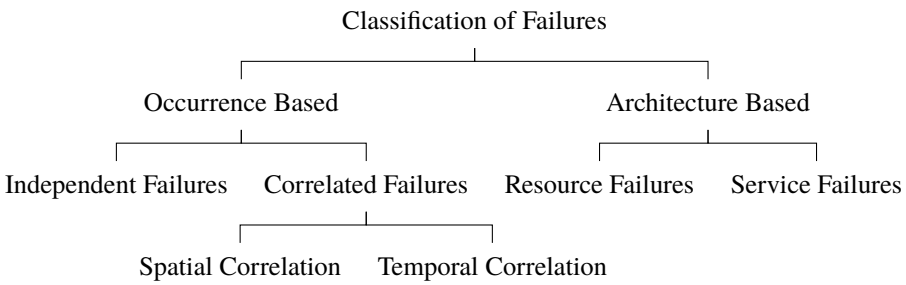


Figure 2.1: Classification of Failures

2.2.1 Classification of Failures

Based on the characteristics of the failures in cloud computing, two different classes of failures: architecture based and occurrence based are formed (Figure 2.1). In the architecture based classification, the failures are further divided into two categories, Resource Failure and Service Failure. As name implies, resource failure is caused by the outage of some physical resources like system breakdown, network or power outage, software error etc. Most of the work on the failure tolerance in the literature has focused on resource failures [78][54][119][157]. Resource failures could occur at the provider or the client end. Service failure in cloud computing means that the cloud provider is unable to provide or the user is unable to get the services promised in the service level agreements (SLAs). Resource failure could lead to a service failure but service could fail even in the presence of working resources during peak loads (section 2.2.2).

The occurrence based classification of failures is all about the interconnection between the failures, whether or not the occurrence of one failure leads to the occurrence of another in the system. Occurrence based failures are further divided into two categories independent failures and correlated failures. Independent failures occur discretely. This type of occurrence is hypothetical because the literature has demonstrated that there is a correlation between failures [55][56][166][134]. In correlated failures, the occurrence of a failure leads to the occurrence of other failures in the system. The failures could be correlated in two different ways: spatial correlation and temporal correlation. A complete survey about the correlated failures is discussed in section 2.2.3.

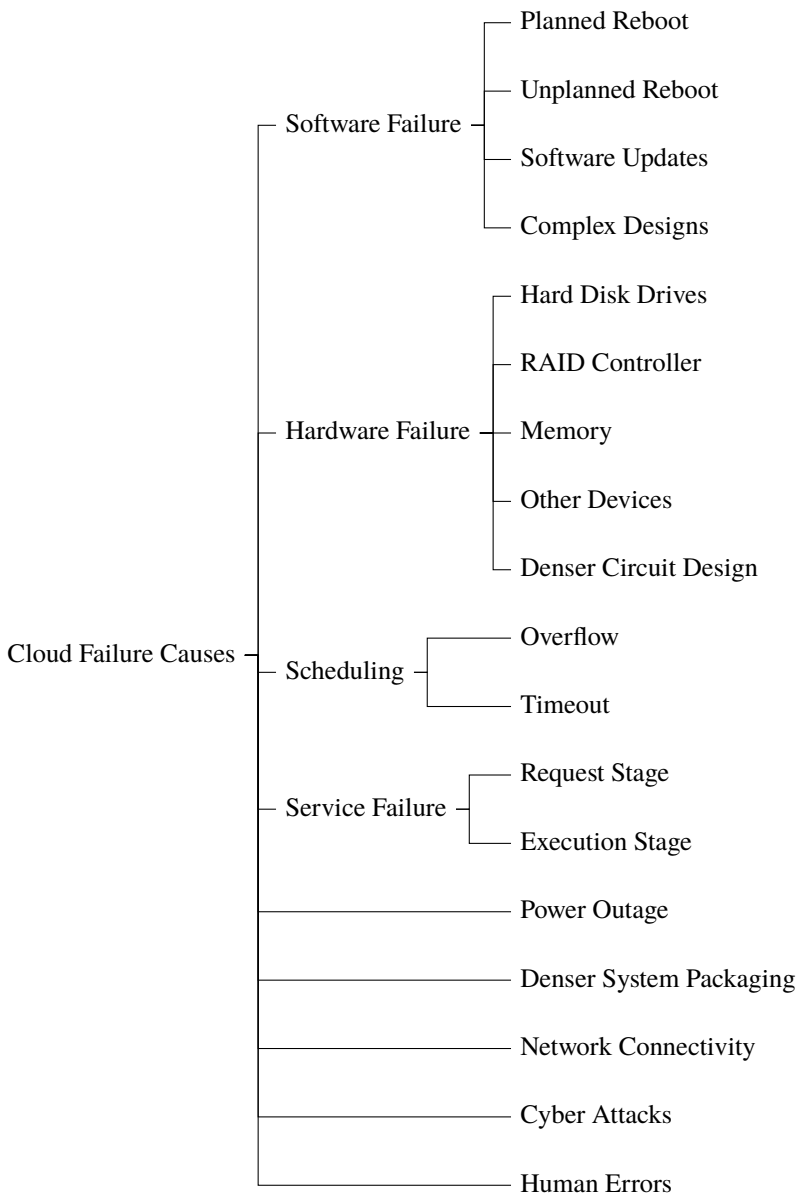


Figure 2.2: Causes of Failures in Cloud Computing

2.2.2 Causes of Failures

To make cloud computing systems more reliable and available all the time, it is very important to understand the causes of the occurrence of the failures. Various causes of failures in cloud computing are given below in Figure 2.2.

Software Failure

As software systems and applications are getting complex day by day, they became a significant reason of system breakdown which causes loss in business and revenue. In October 2013, a cloud based automatic stock trading software of Knight Capital went down for 45 minutes because of an error in trading algorithm which costed $440 million to the company [21]. Sometimes an unexpected error occurs during the process of updating the software, causing the whole system to crash down. In 2013, Azure cloud services of Microsoft were interrupted for 16 hours. It was revealed that they were performing a regular process of updating the firmware in a physical region of the data centers. Something went wrong, which brought down the whole system [43]. Another major service outage was seen in January 2015 for 20 minutes, in which Yahoo Inc. and Microsoft search engine, Bing, went down during the code update [117]. After the crash, the roll back mechanism of Microsoft did not work, which forced the service to shut down from the linked servers to get the point where the system was operating correctly. After a successful update or due to the system maintenance, sometime reboots are scheduled by the service provider about which the service users are informed in advance. Most of the times during planned reboots, service providers consider some backup measures to provide an uninterruptible service to users. On the other hand, unplanned reboots happen after inconsistency in data integration after software or hardware update and the average cost of an unplanned reboot is $9000 per minute. According to Brian Proffitt, up to 20% of attempts are failing in the deployment of software as a service due to the problem of data integration [122]. So it is important to shift application design paradigms from machine-based architecture to cloud-based architectures. Some of the other causes of system failure or performance degradation due to the software failures are memory leakage, un-terminated threads, data corruption, storage space fragmentation and de-fragmentation [152].

Hardware Failure

Hardware failure represents around 4% of all the failures occurred in cloud based data centers. Among all the hardware failures/replacements, 78% are hard disk drives (Figure 2.3)[157]. In

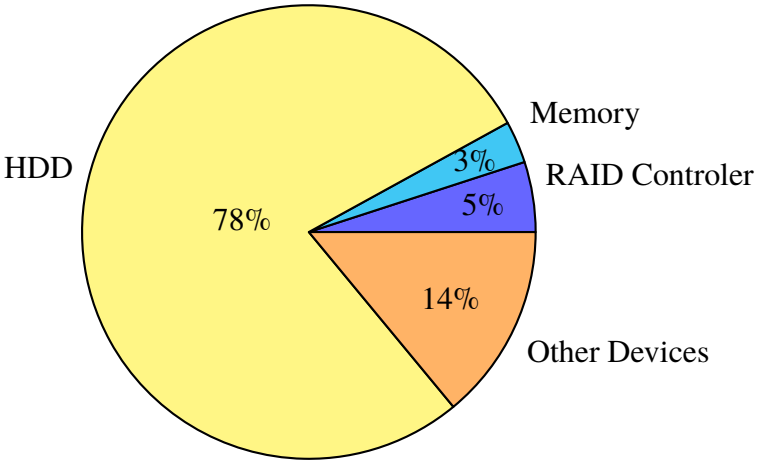


Figure 2.3: Percentages of Hardware Component Failures

2007, hard disk drives and memory modules were the two most common hardware components sent by Google for repair [13]. Hard disk failures increases as the size and age of the clusters increase. K.V. Vishwanath et al. [157], has shown that with age, failure in hard disk drives (HDD) grows exponentially, but after a saturation point it becomes stable. HDD failures can be reduced by timely replacement, and an increase in system reliability will result.

Scheduling

In the cloud computing architecture, schedulers are responsible for scheduling the requests on the provisioned resources meeting the user requirements. Requests waiting to get scheduled are initially placed in an input queue. Being a restricted data structure each queue has a limitation to store a specific number of requests. Exceeding the number of requests than the length of queue will cause drop of new requests and service will be unavailable to the users. This is called an overflow failure. To avoid the overflow of queues, a time-out value is assigned to each request. If the request waiting time in the queue exceeds the specified time out value, then the request will be dropped from the one to make way for fresh requests. This is called time-out failure. This will lead to the service outage in terms of SLA violation due to the delay in cloud computing services.

Service Failure

In cloud computing systems, service failure can happen with or without resource failure. As stated by Yuan-Shun Da et al. [36], the cause of the cloud service failure depends upon the stage of the submitted job such that request stage and executing stage. During the request stage, all the requests with service requirements submitted by users are kept in the ready queue. During this stage, users may not be able to access the services because of overflow or time-out that happens due to overloading of resources such that during peak hours. In such case, the underlying resources are working fine but they are unable to accommodate more requests and service failure happens. Whereas, at execution stage, requests are submitted to underlying physical resources. If services get interrupted, it means the cause of service failure is the outage of resources.

Power Outage

In cloud based data centres, about 33% of the service degradation happens due to the power outage which occurs because of natural disasters or war zones. In 2012, out of 27 major outages of cloud computing services, 6 were caused by the hurricane Sandy alone [23]. In 2011, massive tsunami in Japan put the whole country in power crisis for a long time, and all the consumer services were affected. It is estimated that natural disasters contribute around 22% in cloud computing service outage. An another major cause of power outage is UPS system failures, which contributes 25% of total power outage failures and cost around $1000 per incident.

Denser System Packaging

The infrastructure built ten years ago is now outdated because the data generation has increased exponentially. As the data generation grows by such magnitude, the requirement of storage infrastructure also becomes much higher, triggering the rapid expansion of hardware infrastructure. Designers have begun to design very dense servers like blade servers to keep the space occupied by hardware infrastructure low. Such practice brought the reduction of 65% in total floor space required to set-up an IT infrastructure but increases the device density per square feet. As a result,

heat release by computing components is increased, which further causes a rise in temperature and affects the working of devices. Facebook has revealed that by packing the machines densely, electrical current began to overheat and melt Ethernet sockets and other crucial components. In 2013, data centers of Microsoft faced a severe outage of 16 hours due to overheating of devices that affected its cloud services including Outlook, Hotmail, SkyDrive and Microsoft image sharing service [82].

Network Infrastructure

In distributed computing architecture, specifically in the case of cloud computing, all the services are provided by communication networks. The whole information has been stored and exchanged between servers by using the networks. The outage of the underlying network results in the outage of the services of a cloud computing system. For few cloud based applications such as real time applications, performance of networks plays a key role. A small increment in the network delay can be termed as an SLA violation which will be considered as a service failure. The network services could be broken physically or logically. Around 3% of the service failures in cloud computing systems happened due to the loss of network connectivity. There are various challenges corresponding to the networks such as hop count, bandwidth and encryption that need to be taken care of to make cloud computing services reliable.

Cyber Attacks

Cyber attacks are the fastest growing reason of data center outages. According to Poneman Institute report, the percentage of data center outages due to cyber attacks was 2% in 2010, which is increased to 18% by 2013 and 22% by 2016 [73]. The average downtime cost of outage by cyber attacks is $822,000. An another report on cyber security intelligence published by Ponemon Institute in 2018, has argued that 55% of cyber crimes or threats were from people having access to organization's systems, such that employs [74]. Among other technical issues such as trojan attacks and software loopholes, social engineering [2] is also a major cause of cyber attacks. In social engineering,

attackers play with human psyche by exploiting them with emotions, fear and greed and manipulate them to leak the confidential information.

Human Errors

Along with cyber attacks, human errors also has a big weight (22%) for the causes of failures in cloud computing systems with average cost of $489 per incident. It has been argued that the lack of experience is a main reason of occurrence of human errors. In a study [134], it has been seen that the proportion of human errors is higher during the initial days of deployment of infrastructure. This clearly shows that administrators gains more experience with the time, which reduces the occurrence of human errors. Similar to cyber attacks, social engineering is also a reason for human errors.

2.2.3 Failure Correlation

Correlation is all about the interdependency of activities. If a failure has happened in a part of the system that leads to failures in other parts of the system then it can be said that there is some correlation between the failures. In distributed computing systems such as clouds and grids, if multiple computing components are affected by a common failure then that set or group of computing components is called a shared risk group or shared risk domain because they share a common failure risk [118] just like a communication medium in bus network topology. If the communication medium breaks down then all the data transfer between the nodes using same communication medium goes down. Earlier, most of the research to make cloud environments reliable has been done by considering the independent occurrence of failures [110], which makes the evaluation simpler but error prone in practice. It has been argued that a single faulty node can influence working of the whole system [163]. Even the co-occurrence of failures reduces the effectiveness of various fault tolerance mechanisms such as encoding schemes, replication and backups [127]. Failure correlation can be based on time (temporal correlation) or space (spatial correlation).

Space Correlated Failures

Failures are called spatially-correlated if occurs within a short time interval on different nodes of the same system. Occurrence of failures in a failure burst could be correlated in space. To prove the correlation between the failures in space, general numerical methods such as statistical cluster analysis are required. As a result, Matthieu Gallet et al. [56], proposed a numerical method or model based on three log-normal distribution based aspects such that downtime due to failures, group arrival and group size so as to find the space-correlation between failures occurring during short time intervals. In the given model, a moving window based method is used to find the correlation between failures in a real failure traces. It is found that seven traces out of fifteen shows a strong correlation between the occurrence of failures which has challenged the assumption of independent occurrence of failures.

Temporal Correlated Failures

Temporal correlation is about finding the periodicity in the patterns of occurrence of failures. One of the best methods to find temporal correlation is Auto-Correlation Function (ACF). Ramendra K. Sahoo et al. [127], have identified that only small number of nodes (less than 4%) are prone to 70% of the failures occurred in the system. They also found a strong time varying failure correlation in occurrence of failures on these nodes. Nezih Yigitbasi et al. [166], measured the degree of correlation of failure information gathered from various failure traces with different time lags by using an ACF. In their work, they shifted the plot generated from the failure information according to different lags such as hours, days and weeks to find a repeated pattern. Authors have proposed a formal method to identify the temporally correlated periods that are responsible for downtime of the system.

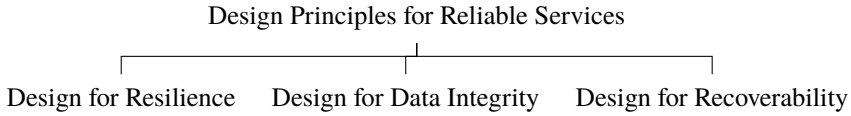


Figure 2.4: Design Principles for Reliable Cloud Computing Services

2.3 Reliable Cloud Computing Services

Reliability in cloud computing is how consistently a cloud computing system is able to provide its services without interruption and failure. Generally the reliability is defined as

The ability of an item to perform a required function under stated conditions for a stated time period. [1]

Cloud computing is a service-oriented architecture so the attributes of the reliability rely on service models such as, Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). To make cloud services reliable, both service providers and service users have their own responsibilities that vary according to the service model. To avoid service failure and to provide resilience, three different design principles for reliable services (Figure 2.4) are proposed by Microsoft Corporation [111]. Providing cloud services while following the given principles will minimize the impact of failures and enhance system resilience so that there is minimal interruption to services. If a failure event occurs at a particular instance, then partial or even delayed services should to be delivered rather than a complete outage. Meanwhile, important measures to recover the service from the degradation or failure should be taken with minimum human intervention. In order to perform the service recovery from a failure, various mechanisms such as checkpointing, replication and VM migration (subsection 2.3.1) have been proposed. During the event of failure and process of recovery from the failure, data integration is a big concern. Moreover, data security is also an issue in these days. There are various incidents in history such as the Sony pictures entertainment hack [39], Dropbox leakage [53] and iCloud leakage [141] that highlights the need to preserve the integrity of the data to make the services reliable and trustworthy.

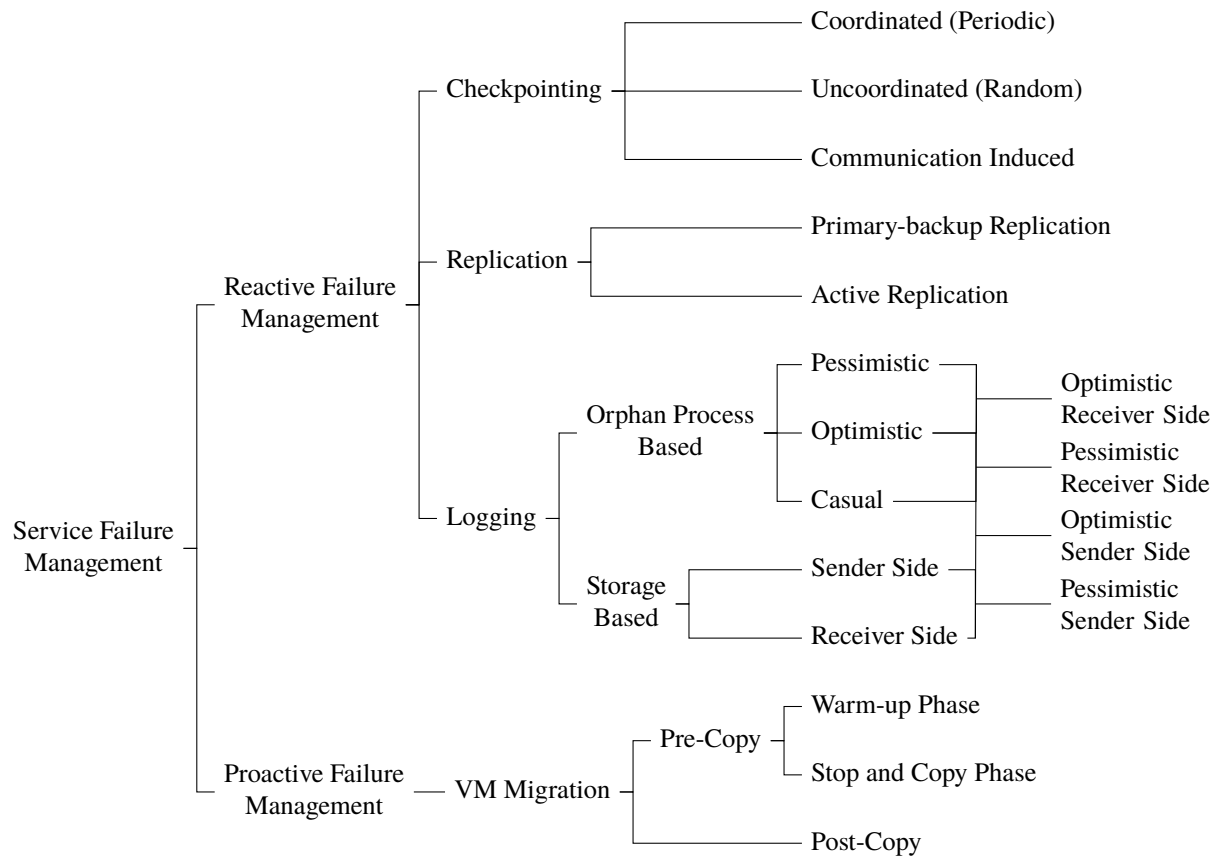


Figure 2.5: Failure Management or Fault Tolerance Mechanisms in Cloud Computing

2.3.1 Service Failure Management in Cloud Computing

To provide reliable services in cloud computing, one needs to manage service failures. All the proposed architectures and techniques designed for well-behaving cloud environment have to be redesigned for a failure-prone cloud environment. To manage resource failures in computing environment for reliability assurance, various techniques and methods have been proposed and implemented (Table 2.1). Since the service-oriented architecture is used by cloud computing, all the techniques and methods need to be explored from the perspective of service reliability. All the failure management techniques are categorized into two groups (Figure 2.5).

Reactive Failure Management

In reactive failure management, measures are taken after the occurrence of a failure. The working of reactive failure management techniques is similar to the working of reactive routing protocols in networks [136]. In reactive routing protocols, there are no routing tables. All the routes are created on demand. In the same way, whenever the failures occurs in cloud services, the required measures are taken by restarting the services from the last execution instance recorded earlier using checkpointing or logging.

Checkpointing is a widely adopted reactive fault tolerance technique, in which the current state of a running process is saved on some backup resources and on the occurrence of a failure, the process restarts or rolls back to the last saved checkpoint. It is shown that the systems running without checkpointing take exponential time to complete the task [44]. By using checkpointing, the exponential time becomes linear. On the basis of the working principle, checkpointing has divided into three different categories: Uncoordinated Checkpointing (Random Checkpointing), Coordinated Checkpointing (Periodic Checkpointing) and Communication Induced Checkpointing [48] (Figure 2.5). Various cloud management software suits such as UniCloud by Oracle, Intel's Data Center Manager (DCM) are incorporated with the checkpointing mechanism to provide uninterruptible cloud computing services. It has been argued that in the large-scale systems like clouds, checkpointing mechanisms could create large overheads as well, if performed frequently [54]. It has been estimated that the checkpointing creates overhead of 151 hours for a job of 100 hours in the petaflop systems [119]. However, if a running program checkpointed infrequently after long intervals, then it will make the re-execution of program lengthy after the failure, which will increase the total execution time of the program. The problem of determining the intervals for checkpointing is called optimal checkpoint interval problem. In the literature, finding the optimal checkpointing interval attracts many researchers [91][37].

Replication is another reactive method to provide fault tolerance in which the backup resources are used to run replicas of the running processes. On the basis of updating of running replicas to handle the data inconsistency, replication is divided into two categories: Primary Backup (Passive)

replication and Active replication(Figure 2.5). Various cloud computing service providers use replication mechanism to provide fault tolerance at different levels. Microsoft Azure uses VM replication to provide fault tolerance at the cloud level. In the case of a failure, Azure always keeps replicated VMs to take charge of a failed VM. At Infrastructure as a Service (IaaS) level, OpenStack, an open-source cloud computing platform, uses data replication to store data by writing the files and objects at multiple disks spread throughout the servers in the data centers. There are many more examples where the replication is in use like DFS replication, Apache Hadoop, Amazon EBS etc. A complete survey of replication mechanisms is done by Rachid Guerraoui et al. [67]. The biggest challenge to run the replicas of a process is to maintain the consistency between the replicas and propagation of update messages. Various methods and mechanisms to handle the challenges and use of replicas in cloud computing environment can be seen in Table 2.1.

Logging or message logging protocol saves or records each process in its present state and messages are saved periodically as the logs at some stable storage. When a process crashes, a new process is created on the place of a crashed process by using the recorded logs. To get the pre-failure state of a crashed process, all the logged messages are evaluated in the same order in which they were generated. Once the new process is created after a crash, the state of the new process needs to be consistent with other running processes. If the state of the process remains inconsistent then the process will be known as an orphan process. To reduce the overhead of logging, checkpointing is incorporated with logging (Table 2.1). Once the checkpoint is saved for the state of a process then all the logged messages before the checkpoint can be removed to save storage space. We classify the process of logging into two classes: Orphan process based and Storage based. These are further combined with each other to make more classifications [108] (Figure 2.5). In the upper sections, various coordinated methods used to provide fault tolerance in cloud computing systems are discussed. However, because of the overheads generated by the coordination between the processes, they have scalability issues. The uncoordinated methods such as message logging seems to be a good option in terms of application makespan for cloud computing systems. Pierre Lemarinier et al. [96], have shown that if the mean time between failures (MTBF) is less than 9 hours then messaging logging is a better option than the coordinated checkpointing because of less

overheads.

Proactive Failure Management

Due to the large overheads and expensive implementation of reactive failure management mechanisms, cloud service providers have begun to adopt proactive failure management mechanisms. In proactive failure management, the prevention measures get taken before the occurrence of failure. The productivity of proactive failure management methods depends upon the accuracy of adopted failure prediction mechanisms [55] [76]. On the basis of the failure prediction results, the running processes migrates from the suspected resources to healthy resources for an uninterruptible execution. The accurate failure prediction will make the failure management more efficient and reliable. Failure prediction is classified into two categories: offline failure prediction and online failure prediction. A complete survey about the failure prediction methods is done by Felix Salfner et al. [130]. After the results of the failure prediction methods, suitable actions are taken by proactive fault tolerance mechanisms. Migration is the method that is used to provide fault tolerance by incorporating failure prediction mechanisms. With the introduction of high speed networks and distributed architecture of computing, the migration of running tasks or VMs became possible. In cloud computing systems, migration is divided into process migration [113] and VM migration. By considering the dynamic nature of the cloud infrastructure, only VM migration based fault-tolerance methods are considered in Table 2.1. To migrate the running VMs from a faulty server to a healthy one, two methods have been proposed in the literature: Pre-Copy and Post-Copy (Figure 2.5).

Pre-copy VM Migration Approach has two different phases: Warm-up Phase and Stop-and-copy Phase [139]. In warm up phase, hypervisor copies the state of running VMs such as CPU state, memory state, and state of other devices from a faulty server to the destination server. As the warm-up phase completes, the VM stops at the source machine and stop and copy phase initiates. The stop and copy phase copies the remaining files or pages (if any) in the memory that gets modified (dirty pages) during the warm-up phase. After the transfer of all the pages, the VM resumes its

execution over the destination machine. The time between suspension of a VM from the source node and resumption over the destination node is called down-time. Many of the hypervisors such as VMware, Xen and KVM use pre-copy migration approach [102].

Post-copy VM Migration Approach is used when the running VMs get suspended at the source node and migrates to the destination node with partial attributes of the execution state such that CPU state and register usage [71]. After getting the destination node, VMs resume with the execution. In parallel, the source machine also stays active serving the migrated VMs. Whenever a VM do not find a page in its local memory, it generates a page fault (network fault). On the generation of a page fault, destination machine redirects the page request to the source machine, which in-turn responds with the faulted page. In general, the memory image can be transferred in the background after execution of VM starts at destination or it can be transferred on-demand in response to page fault.

As stated earlier, along with providing reliability to the services and optimized resource utilization, VM migration has also been proved as a very promising technique to manage the energy consumption in cloud computing systems. Thorough details about the mechanisms used to manage the energy consumption in cloud computing paradigm are discussed in the next section.

2.4 Energy Management in Cloud Computing

Besides reliability of cloud computing services, energy consumption by the underlying complex infrastructure providing cloud services is also a big concern for cloud service providers. As increasing the reliability of cloud services makes it profitable by attracting more users or clients, decrease in the energy consumption will make it even more profitable by reducing the operational expenses of underlying infrastructure in terms of electricity bills. Besides the construction of data centers by adding temperature monitoring equipments, optimized air vent tiles, putting plates to block cold air passing through the racks, designing of optimized software systems is also very important for the proper utilization of resources of cloud infrastructure to increase the energy efficiency. As shown in Figure 2.6, energy consumption can be optimized at the hardware level,

Table 2.1: Survey of Failure Management Mechanisms in Cloud Computing

Authors	Service Failure Management	Failure Management Method	Objectives	Architecture	Workload
Bakhta Meroufel et al.[107]	Reactive	Checkpointing	System Availability	Cloud	Communication Induced
Song Fu et al. [54]	Proactive	VM Migration	System Availability, Resource Utilization	Virtualised Clusters	HPC
John Paul Walters et al.[161]	Reactive	Checkpointing, Replication	System Availability, Resource Utilization	Virtualised Clusters	HPC
Rachid Guerraoui et al.[67]	Reactive	Replication	System Availability, Resource Utilization	Clouds	Not Specified
Ravi Jawahar et al. [81]	Proactive	VM Migration	System Availability, Resource Utilization	Clouds	Multiple
Aiqiang Gao et al.[59]	Reactive	Replication	System Availability	Cloud	Web
Da-Wei Sun et al.[145]	Reactive	Data Replication	System Availability, Resource Utilization	Cloud	Not Specified
Anju Bala et al. [11]	Proactive	VM Migration	System Availability	Cloud	Scientific
Nicolas Bonvin et al.[17]	Reactive	VM Replication	System Availability, Resource Utilization	Cloud	Not Specified
Haikun Liu et al. [101]	Reactive, Proactive	Checkpointing, VM Migration	System Availability, Resource Utilization	Virtualized Clusters	Web
Mohammed A. Alzain et al.[5]	Reactive	Replication	System Availability	System Security, Resource Utilization	Multi-Clouds
Xianqing Yu et al.[168]	Reactive	Replication	System Availability, System Security	Public Cloud	Not Specified
Tung Nguyen et al.[115]	Reactive	Replication	System Availability, Resource Utilization	Cloud	Multiple
Brendan Cully et al.[35]	Reactive	Checkpointing, Replication	System Availability	Virtualised Systems	Web
Daeyong Jung et al. [83]	Proactive	VM Migration	System Availability, Resource Utilization	Cloud	Multiple
Bahman Javadi et al. [79]	Reactive	Checkpointing	System Availability, Resource Utilization	Virtualised Clusters	Parallel
Bahman Javadi et al. [78]	Reactive	Checkpointing	System Availability, Resource Utilization	Hybrid Clouds	HPC
William Voorsluys et al. [159]	Reactive, Proactive	Checkpointing, VM Migration	System Availability, Resource Utilization	Cloud	Compute-Intensive
Lin Yao et at. [165]	Proactive	VM Migration	System Availability, Resource Utilization	Cloud	Not Specified
Ao Zhou et at. [173]	Proactive	VM Replication	Service Availability	Cloud	Not Specified

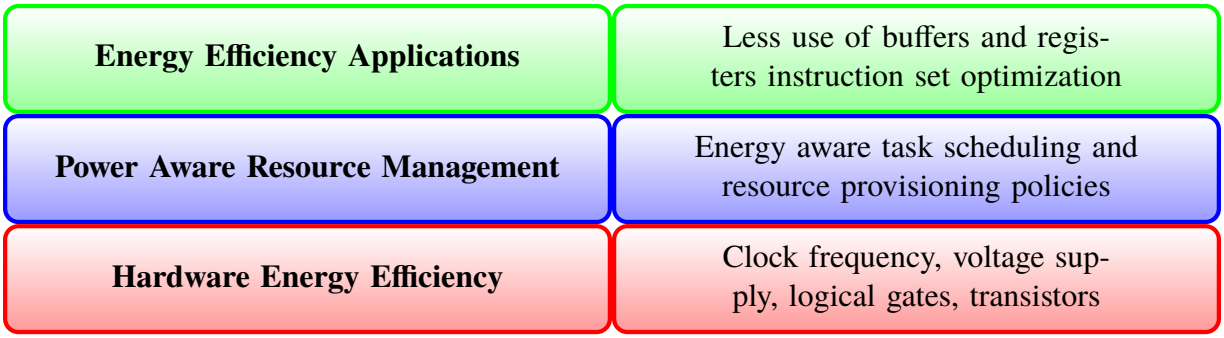


Figure 2.6: Levels of Energy Efficiency Enhancement and Possible Solutions

software level and intermediate level. In the following sections, different techniques and methods to regulate the energy consumption in cloud computing systems are explored. A complete list of the existing most energy efficient distributed computing systems is provided by Green500[2].

In some studies, problems of high power consumption and high energy consumption are considered separately [15]. But because of direct proportional relation between the energy and power consumption (Equation 2.1), both energy and power are used interchangeably in this study and this has been done by many studies in this domain [51].

$$E = PT \tag{2.1}$$

2.4.1 Static Power Management

Also known as offline energy management deals more with circuitry systems. It is more engineering oriented approach. In static management of power, whole optimization takes place at the system level during the design time. It deals with the geographical distribution of the processing centers, circuit manipulation, redesigning of architectures, instruction sets, transistor sizing, path balancing and factorization[41]. The main goal of the static power management is to keep the energy consumption or power consumption low by using low power usage components. In this category, the energy consumption is managed at two levels: CPU level and System level. It is shown

[2]http://www.green500.org/greenlists

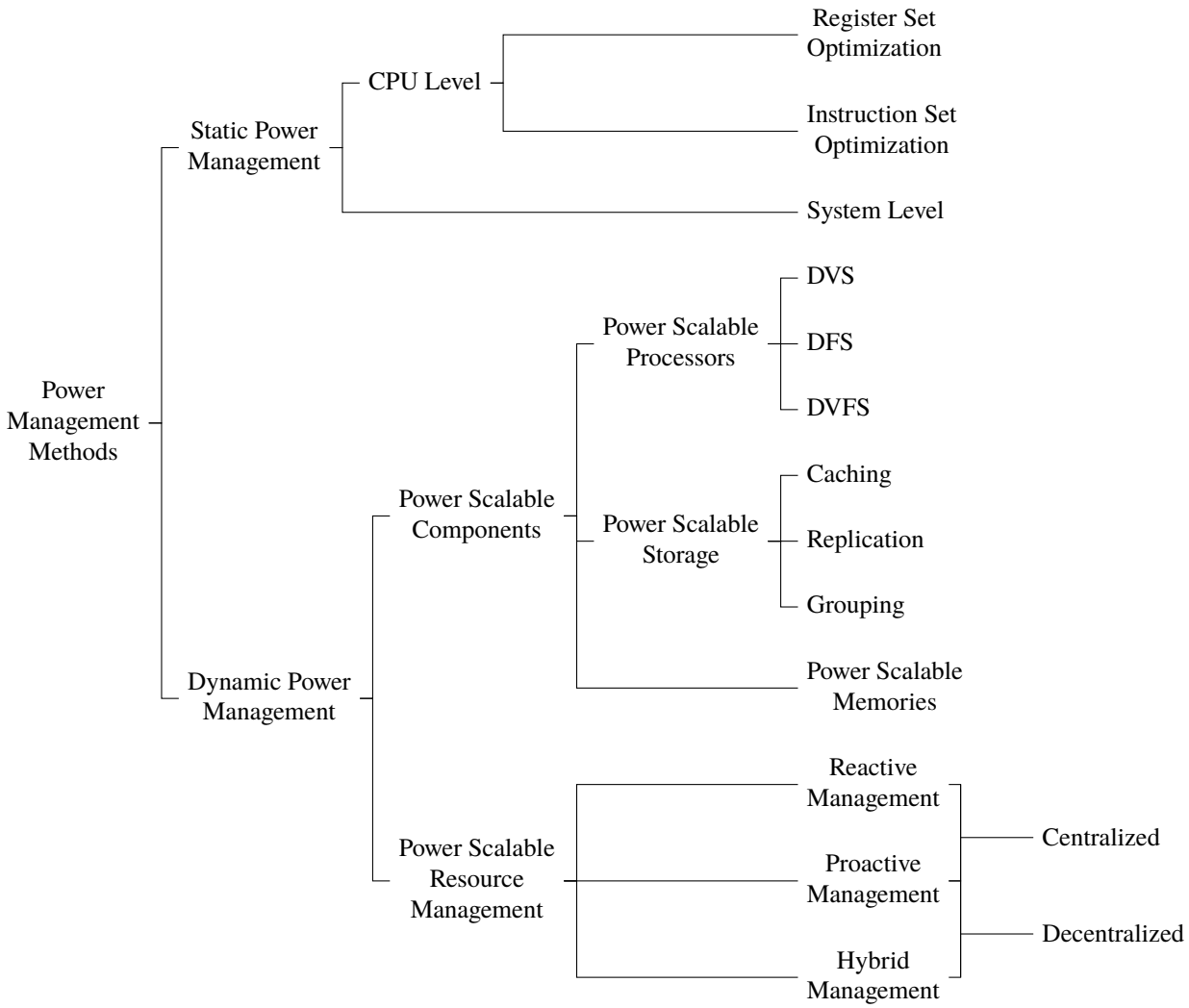


Figure 2.7: Energy/Power Management Mechanisms in Cloud Computing

that among all the computing components, CPU consumes 35-50% of total system energy and provides a big scope to optimize energy consumption [153]. At CPU level, the optimization can be done at register level or at instruction set level. At register level, all the measures to reduce energy/power consumption takes place by optimizing the register transfer level (RTL) activities and at the instruction set level, different types of instruction set architectures (ISA) have been proposed to reduce the power consumption such as reduced bit-width ISA.

Along with CPU, there are other components which are big contributors to the overall power consumption of the system such as memory components, network facility and software systems. System level static power management methods have been used to regulate the energy/power consumption by such components. System level power optimization also deals with system set-up or deployment techniques. Questions such as how to choose the right components during the set-up phase of cloud systems to minimize the asynchronization between different components, how to place the servers to minimize the delays, choice of operating systems and application softwares are answered using system level power management methods. Architectures such as FAWN [8] and Gordon [29] are proposed to couple the low power CPUs with local flash storage and data center power systems to balance the computation and I/O activities to make the cloud computing architectures more performance and energy efficient. Geographic distribution of the machines [149], choosing components with maximum compatibility and network topologies to minimize the power consumption belongs to system level power optimization.

2.4.2 Dynamic Power Management Mechanisms

Dynamic power management (DPM) deals with the regulation of energy consumption by using software based policies. Each type of server components provides a different dynamic power range such as the difference between the maximum power consumption and minimum power consumption. In Figure 2.8, it has shown that CPUs can consume around 30% of their peak power consumption in the low activity modes, which gives the range of 70% to scale up and down. On the other hand, memory and disk drives have the dynamic range of 50% and 25%, respectively

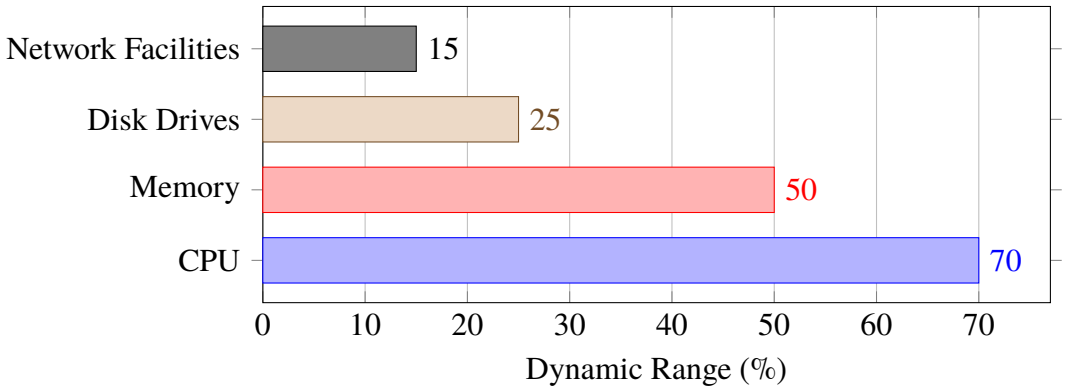


Figure 2.8: Dynamic Range of Power Consumption of Various Server Components

followed by the network facilities such as switches or routers, which have the range of only 15% [15]. On the basis of dynamic range of power consumption, the working of components can be scaled up or scaled down to regulate the power/energy consumption. On the basis of approach used to reduce the power/energy consumption, the classification of DPM methods is done at two levels, Hardware Level (using Power-scalable components) and Software Level (using Power-scalable resource management).

Dynamic Power Management using Power-scalable Components

At the component level, all the components supporting low activity modes are considered as the power scalable components (Figure 2.8) and can be manipulated using DPM methods. As stated earlier, CPU is the major power consuming component followed by the memory units. So in majority of cases, DPM methods are found to be focusing on two components such that CPU and memory.

Power Scalable CPUs use the relation between the power supply, operational frequency and supply voltage (Equation 2.2) to regulate the power utilization in CPUs.

$$P_{dynamic} = aCfV^2 \tag{2.2}$$

where, a is the logical or switching activity, C is the capacitance, f is the operational frequency and V is the supply voltage. Advancement in the architectures made CPUs able to run at different

activity modes using different voltage and frequency rates.

In complementary metal oxide semiconductor (CMOS) circuits, the energy consumption increases quadratically as the supply voltage increases. All the above mentioned power management techniques exploit this factor by reducing the supply voltage (DVS), operational frequency (DFS) or both at the same time (DVFS) [92] [174]. There are many ways to scale down the high voltage supply to decrease the energy consumption but one of the best is to exploit the stall time. Due to the speed gap between the main memory and the processor, significant amount of clock speed of processor get wasted whilst waiting to get the required data from the main memory. During the waiting time (stall time), the processor frequency can be brought down by manipulating the supply voltage to save the excessive energy/power consumption [89]. Many semiconductor chip makers are using such voltage and frequency scaling techniques at different levels and in different devices. Intel's Woodcrest Xeon Processors works at eight different operating frequencies by reducing the maximum operational frequency by 8.3%, 16.5%, 25%, 33.3%, 41.6%, 50.0%, 58.3%, 66.6%, 77.7%, 88.9% and 100% [58]. By using CPU throttling, Intel has developed SpeedStep CPU throttling technology and AMD has developed two CPU throttling technologies: Cool'n'Quiet and PowerNow!. Along with the frequency scaling of the CPU's, AMD has also adopted frequency throttling in Graphical Processing Units (GPU) as AMD PowerTune and AMD ZeroCore Power.

Power Scalable Storage Systems regulate the activity of storage devices such as disk drives to reduce power consumption. In distributed computing systems, energy consumption by disk drives is significant. It has been estimated that around one-third of the total electricity supplied to the data centers is required for the mechanical operations of disk storage systems [87]. Typically, when a disk is in standby, it consumes about one tenth of the power that it consumes during the spinning mode. The energy consumption by storage systems in large data centers need to be considered seriously because the requirement of the storage systems is increasing by 60% annually [120]. In large cloud based data centers, disk drives usually remains underutilized and use less than 25% of their total storage capacity. This provides large scope to reduce the energy consumption by disk drives by increasing the utilization and by turning off the unnecessary disks [68]. Various methods to make storage system power efficient are given in Figure 2.7. A thorough survey on the energy

efficiency of the disk drives is done by Tom Bostorn et al. [18].

Power Scalable Memories are addressed least among all the components to minimize the energy consumption in large scale distributed computing systems. According to David Howard et al. [38], under specific workloads, memory unit can consume 23% on average, of the total power consumption. In Figure 2.8, the dynamic range of power consumption of memories is 50%, which provides plenty of scope to increase the power/energy efficiency of memory units. Like CPUs, the concept of low frequency and less voltage for power reduction (DVFS) is applicable to memory units also. In the case of DRAMs, the power consumption of some of the components such as storage arrays of DRAM can be scaled by V and some of the components can be scaled by V^2. Making energy aware memory components and using them in cloud computing environment gives rise to new challenges. Making power efficient memories can be achieved at the price of performance. The power aware techniques used in the memory units should be leveraged to save overall power consumption of the large scale systems without effecting the performance of the systems. In response to this, a software platform called Memory Management Infrastructure for Energy Reduction (Memory MISER) consists of a modified Linux kernel and implementation of a PID controller is proposed by Matthew R. Tolentino et al. [150]. The proposed architecture has been proved to reduce energy consumption of memories by up to 70% and up to 30% for the overall system.

Dynamic Power Management using Power-scalable Resource Management

With the adoption of energy efficient components in the infrastructure of cloud computing systems and, due to the vast amount of data for processing, the management and monitoring of the resources is very important. Wise management of the resources including resource provisioning, task scheduling, performance monitoring leads to less energy consumption and more profit-aware computing. Although the management of the resources is a general term for the distributed computing environment but in the context of cloud computing it is more associated with the technology of Virtualization. The employment of Virtualization makes it possible to minimize the number of

working resources by keeping the utilization of resources high by executing more VMs processing different workloads.

In this section, various mechanisms that execute the tasks on cloud computing infrastructure in an energy efficient manner are highlighted. This section answers several questions such as how to provision the resources in an energy aware manner, How to distribute or schedule the workload among the provisioned computing components in an energy efficient way, When to migrate the running tasks from one underutilized resource to other to save the power consumption, When and how many computing components need to be turned on or turned off to save energy. In the literature, many algorithms, heuristics or architectures are proposed to handle the issues of power/energy consumption in cloud computing environments (Table 2.2). Mechanisms to reduce the energy consumption by using software techniques are divided into three categories: Reactive, Proactive and Hybrid [70]. These can be implemented in centralized and decentralized ways.

Reactive Management of Resources takes all the measures to manage the energy consumption according to the current state of the system. The reactive mechanisms are based on real time system monitoring or feedback. The continuous monitoring of the system takes place and according to the pre-defined constraints such as thresholds, the corrective actions gets taken by migrating or consolidating the workload to regulate the energy consumption of the system. The productivity of reactive management of energy consumption depends upon the accuracy of the monitoring procedure. In virtualized computing environments like clouds, when the resources are not fully utilized the migration or consolidation of the running VMs to some other resource is possible and promised as the best technique to reduce the energy consumption. Along with regulating the energy consumption, with the efficient utilization of resources, the carbon emission rate is also a concern. With the increase in energy consumption, heat dissipation by computing infrastructure also increases which adds to the temperature of computing components and increases their carbon footprint. Such rise in temperature increases the requirement of cooling infrastructure and corresponding energy consumption to keep the temperature of computing infrastructure low in order to avoid any break down. Exploitation of reactive energy consumption methods can help to reduce the cooling requirements by keeping the number of running/active resources less, which further reduce the

total energy energy consumption and results in less emission of carbon content to the environment. A thorough survey of work on the energy consumption by using the reactive resource management mechanisms is given in Table 2.2.

Proactive Management for Resources also known as predictive management of resources use the information about the average behaviour of the system rather than the current state of the system. The decision about choosing the optimized resources in terms of performance, energy consumption and reliability are taken on the basis of the data collected during the previous runs. By using the collected data, predictions are done about the behaviour of the system to make the adequate decisions about the allocation of resources to minimize the energy consumption. In the literature various prediction models are proposed to minimize the energy consumption [19]. Similar methods using predictive approach to reduce energy utilization in cloud computing environment are presented in Table 2.2.

Hybrid Management of Resources use both predictive behaviour of proactive methods and monitoring behaviour of reactive methods to tune the energy consumption and resource utilization. Due to the dependency on the results of prediction mechanisms, methods in proactive resource management always lags because as mentioned in subsection 2.3.1, it is hard to predict the behaviour of the system accurately including the energy consumption. However, due to the large overhead, the reactive energy efficiency resource management methods possess delays, which add to the power inefficiency of the whole system. By combining the merits of reactive and proactive methods, the hybrid methods are designed. In the literature (Table 2.2), some work is done by various authors combining both reactive and proactive methods to reduce the energy consumption of the cloud computing systems.

Table 2.2: Survey of Energy Management Mechanisms in Cloud Computing

Authors	Power Scalable Resource Management	Objectives	Workload	Components	Power Saving Method
Anton Beloglazov et al. [14]	Reactive	Resource Utilization	Web	CPU	DVFS
Saurabh Kumar Garg et al. [61]	Reactive	Carbon Footprint, Profit	HPC	CPU	DVFS
Jennifer Burge et al. [24]	Reactive, Proactive	Profit	Multiple	System	On/Off
Yiyu Chen et al. [30]	Reactive, Proactive Hybrid	Profit	Web	CPU	DVS
Young Choon Lee et al. [94]	Reactive	Resource Utilization	Multiple	System	
D. J. Bradley et al. [19]	Proactive		Web, Email Database	CPU	On/Off
Chu-Fu Wang et al. [162]	Proactive	Resource Utilization	Multiple	System	On/Off
Mahboobeh Ghorbahi et al. [64]	Proactive		Multiple	CPU, Memory	
Joseph Subirats et al. [143]	Proactive	Resource Utilization	Batch, Web		
Shekhar Srikantaiah et al. [142]	Reactive	Resource Utilization	Multiple	CPU, Disk	
Negin Kord et al. [90]	Proactive	Resource Utilization, Profit	MapReduce	System	
Anshul Gandhi et al. [58]	Proactive	Resource Utilization	Web	CPU, Memory	DFS, DVFS
Ifeanyi P. Egwutuoha et al. [46]	Proactive	Resource Utilization	HPC	CPU	
Trung Le et al. [93]	Reactive	Carbon Footprint, Profit	HPC	CPU	
S. K. Tesfatsion et al. [146]	Reactive	Carbon Footprint	Video Encoding	CPU	VM Migration, DFS
Bharti Wadhwa et al. [160]	Reactive	Resource Utilization, Carbon Footprint	Multiple	CPU	DVFS
Ateíeh Khosravi et al. [86]	Reactive	Carbon Footprint	BoT, Web	System	VM Placement
Laurent Lefevre et al. [95]	Proactive	Carbon Footprint, Profit	Multiple	System	On/Off
Saurabh Kumar Garg et al.[60]	Reactive	Carbon Footprint	HPC	CPU	
M. Mezmaz et al. [109]	Hybrid	Resource Utilization	HPC	CPU	DVS
Anshul Gandhi et al. [57]	Hybrid	Resource Utilization	Web	System	
Riky Subrata et al. [144]	Hybrid	Resource Utilization	Web	System	Hibernation
Akshat Verma et al. [130]	Hybrid	Carbon Footprint	Multiple	CPU	On/Off

2.5 Trade-off between Reliability and Energy Efficiency in Cloud Computing

In the previous sections, it is observed that most of the research has focused either on service reliability or energy efficiency in cloud computing environments. As analysed, existing mechanisms do provide reliability to cloud computing services and have proved to be very efficient and optimized [91][31]. By using these methods, cloud computing service providers are claiming on the one hand that their cloud services are more than 99% available in terms of uptime allowing only 80 hours of downtime per year, approximately. However, all the given methods require extra backup and storage resources to store logs and checkpoints to allow last state system recovery in the case of failure or interruption. Adding extra resources to the infrastructure increases the energy consumption at a greater rate than reliability gains and has a direct impact on the profit margins of the service providers and users and negatively impacts natural environment.

Energy management mechanisms that regulate system performance and hardware resources reduce the system energy consumption. The key techniques used to reduce energy consumption is running the resources on low power scaling level or by turning off the idle resources such as backup, which will reduce the reliability of the system. For example, in the case of VM consolidation (key technique to reduce energy consumption in cloud computing systems), if a physical machine fails due to some hardware or software issues before the completion of tasks and there are no recovery resources, then all the VMs and their corresponding processes will have to start again. This will dramatically increase overheads such as energy consumption and resource utilization. Service providers will lose a lot of revenue in terms of penalties for SLA violations and most importantly, trust of the users.

A crucial trade-off between reliability and energy-efficiency of cloud computing systems is shown in Figure 1.1 of chapter 1. On the one hand, reliability of the system increases as the resource redundancy increases. But increasing the number of redundant resources used to store backups or to run replicas has adverse effect on the energy efficiency of cloud computing systems.

On the other hand, as the frequency of VM consolidation increases, energy efficiency of the system increases. But high VM consolidation has the negative effect on the reliability of the system. Both reliability and energy efficiency of cloud computing systems increases, asymmetrically. This trade-off opens up new opportunities and challenges in cloud computing systems by considering both these elements, simultaneously. It is very important to reach equilibrium between these two metrics from different perspectives such as quality of services, revenue, operational cost and environment. There is a distinct need for more research in the area of optimising the relationship of system reliability and energy efficiency in cloud computing systems (Table 2.3). The following section of this chapter seeks to outline the current research into the interplay of reliability and energy efficiency in cloud computing systems.

2.5.1 State of the Art in Reliability and Energy Efficiency Mechanisms in Cloud Computing

In this section, current research combining reliability and energy efficiency of cloud computing has been highlighted and gaps have been identified. The brief description of this section is provided in Table 2.3.

Hamid Reza Faragardi et al. [51] have proposed an Integer Linear Programming (ILP) based mathematical model to regulate the reliability and energy consumption of the cloud computing systems by taking into consideration quality of services in terms of service deadlines. On the basis of this model, a swarm intelligence resource scheduling method based on Imperialist Competitive Algorithm [10] is proposed to allocate the resources in a failure-aware and energy-efficient way. To introduce the failures in the systems, Faragardi has used a Poisson process-based failure model that generates constant and independent failures. Along with the failure model, an energy model is also proposed based on CPU utilization. By using the equations for reliability and energy consumption, a common ILP-based cost function is used to balance both energy and reliability. The proposed solution has improved the energy utilization and system reliability significantly by 17% and 9% respectively in comparison to a hybrid genetic algorithm.

Table 2.3: Survey of Trade-off between Reliability and Energy Management Mechanisms in Cloud Computing

Authors	Failure Management Method	Failure Management Type	Energy Management Method	Energy Management Type	Failure Type	Performance Evaluation Method	Application Model
Hamid Reza Faragardi et al. [51]	Imperialist Competitive Algorithm based Failure Aware Resource Allocation	Proactive	On/Off	Proactive	Independent Failures	Mathematical Model, Simulation	Independent Tasks
Ifeanyi P. Egwutuoha et al. [46]	Failure Prediction based Process Migration	Proactive	Process Migration	Proactive	Independent Failures	Simulation	Message Passing Interface(MPI) Applications
Altino M. Sampaio et al. [131]	Failure Prediction based VM Migration	Proactive	VM Consolidation On/Off	Proactive	Independent Failures	Simulation, Real Platform Evaluation	Poission Distribution and Google based Workloads
M. el Mehdi DIOURI et al. [104]	Checkpointing, Uncoordinated Message Logging, Process Coordination	Reactive	Voltage Scaling	Reactive	Independent Failures	Benchmarking	High Performance Computing Applications
Longxin Zhang et al. [171]	Reliable Resource Allocation	Reactive	Dynamic Voltage and Frequency Scaling (DVS)	Reactive	Independent Failures	Simulation, Benchmarking	Fast Fourier Transformation, LU-decomposition, Gaussian Elimination
Wei Deng et al. [40]	Genetic Algorithm based Server Cosolidation	Proactive	On/Off	Proactive	Independent Failures	Mathematical Model, Simulation	Light, Normal, Intensive Application Workloads
Jia-Chun Lin et al.[100]	Task Re-execution Policy	Reactive	Load Balancing	Reactive	Independent Failures	Theoretical Simulation	MapReduce Workload

In this study the occurrence of failures are modelled by using Poisson distribution, which is shown to be a poor fit by many researchers [121] [134]. Normal and log-normal distributions have been proved a better fit for failure generation modeling. Authors also modeled independent occurrence of failures, which is challenged [127] [56] by showing temporal and spatial correlation between the failures.

Ifeanyi P. Egwutuoha et al. [46] have developed a generic proactive energy efficient fault tolerance model independent from redundant resources for cloud computing systems executing high performance computing (HPC) applications. To provide immunity from task failures, a rule based prediction mechanism is used to foresee failures using the data gathered by a back end service "FTDeamon" using LM-sensors. A mathematical model is developed to evaluate the weight of the current state by multiplying the LM-sensor values of all components of systems. After calculating the current weight, a comparison is done with the critical state threshold value. On the basis of comparison result, decisions about provisioning of new resources, relinquishing of faulty ones and migration of processes are taken. To make the method less expensive and energy efficient, no extra resources are provisioned initially to provide fault tolerance. On the basis of the results of failure prediction mechanisms, extra resources are provisioned to initialize the VMs to migrate the running processes from failing hosts/resources. Process level migration is used instead of using traditional VM level migration because process level migration has less overheads and makes the migration fast, which further helps to reduce overall energy consumption and make fault tolerance more dynamic and less complex.

The proposed mechanism is designed for message passing interface applications, which require uninterruptible functioning of resources for a long duration. As no backup resources are used to provide immunity from failures to running processes, this algorithm depends highly upon the accuracy of the failure prediction mechanism. The average accuracy of failure prediction mechanisms is 76.5% [54]. This level of accuracy is unsuitable for HPC workload. To make the mechanism more attractive, both reactive and proactive fault tolerance mechanisms should be used simultaneously.

Altino M. Sampaio et al. [131] have proposed two algorithms POwer-and Failure-Aware

Relaxed time Execution (POFARE) and POwer-and Failure-Aware Minimum time Execution (POFAME). They addressed the problem of mapping of VMs to physical machines, so as to increase the completion rate of the tasks with minimum energy consumption in a private cloud computing environment. Stop and copy VM migration employing failure prediction is used to make the services available and to execute the tasks by deadline without any interruption. CPU has chosen optimistically on the basis of predicted Mean Time between Failure (MTBF) and according to the capacity required to finish the tasks within their respective deadline. SLA terms are ensured by completing the tasks on time and avoiding penalties. A tentative private cloud architecture is also designed in which a cloud manager monitors the status of virtual and physical machines. Based on the information, the cloud manager allocates tasks concerning energy consumption improvement, so as to facilitate physical machine fault tolerance. To save energy and provide fault tolerance, VM migration is employed as well as putting free physical machines in sleep mode. Three other algorithms: Common Best-Fit (CBFIT), Optimistic Best-Fit (OBFIT) and Pessimistic Best-Fit (PBFIT) are used to evaluate the performance of proposed algorithms. After the intensive simulations performed by using Poisson distribution-based random workload and Google-based workload, POFARE outperformed all the algorithms and gave the best results.

A limitation of proposed energy model lies in the use of only CPU power consumption, without consideration of any other components such as memory and disk-drives and heterogeneity of physical machines. Voltage scaling would have been more energy efficient solution than entering and waking-up the nodes from the sleep state. Similar to Hamid Reza Faragardi et al. [46], performance degradation of the system is not considered. To make the reliability and failure models simple, most of the researchers assume either the system works fine or it fails. This kind of binary behaviour is valid for components such as CPU but not for the whole system because in virtualized computing environments, system slowdown or performance degradation occurs because of shared resources between VMs. This leads to the system failure. In the given work, running multiple VMs on the same node is implemented but assumed no performance interference, which is not the case in the real world and has to be considered. Failure tolerance in proposed solutions relies completely on failure prediction. If physical machine fails outside of the failure forecasts,

then all the VMs have to be re-initiated because of a non-forecasted failure then all the running VMs have to be re-initiated.

Peter Garraghan et al. [62] have done an empirical analysis by using Google traces to analyse the failure related energy waste in cloud computing environments. This analysis highlights the impact of failures at task level (software level) and server level (hardware failures). All the terminal events taken from Google cluster traces are divided into three categories: Kill, Evict and Task fail. SpecPower2008 benchmark [16] is used to calculate the energy consumption of per failure event. In the study, it is observed that Kill and Evict type of events in the traces contribute to more energy wastage (48% and 39% respectively) than task failures (13%). The occurrence of kill and evict events is considered because of scheduling, which is one of the reasons of failures in cloud computing services (Figure 2.2). All the tasks are assigned with different priorities from 0 to 9 and occurrence of failures is scaled on the basis of priorities of the interrupted tasks. The low priority tasks are found most prone to failures with mean time between failure of only 1 hour and vice versa for high priority tasks (48h and 58h, respectively). 35% of failures occur on tasks with lower priority. At the server level, numbers of failures are calculated on the basis of the architecture type of the underlying servers. The frequency of the occurrence of failures (MTBF) and recovery time (MTTR) is independent of the population of the servers. For energy wastage, priority 0 tasks have only minor energy wastage but priority 8 and 9 tasks waste large amount of energy in comparison to the number of failures because of resubmissions. This means that the energy wastage is independent from the number of task failures. The proportion of energy wastage depends upon the characteristics of the failed tasks such as the task length. The longest running tasks (priority 9) have the greater impact, which wasted a considerable portion of the energy (up to 65%). From the analysis of all type of terminal events, task failures contributed up to 21% of the total energy wastage because of the resubmission and re-computation of failed tasks.

In conclusion, it is claimed that the choice of a mechanism to regulate the energy wastage in the presence of failures should be made by considering physical architecture or scenario. Inappropriate mechanism will lead to more energy wastage rather than reduction, for example the adoption of task migration for low priority tasks will lead to high increase in execution time, which further

increase the energy consumption. In the given work, the methodology to carry out the analysis is purely empirical and missing a mathematical model or formal procedure to regulate the energy consumption in the presence of failures.

M. el Mehdi DIOURI et al. [104] have evaluated the energy consumption while using coordinated and uncoordinated fault tolerance protocols. As uncoordinated protocol, message logging is adopted in which logs are stored at Hard Disk Drive (HDD) or Random Access Memory (RAM). When comparison is made between the power consumption of RAM logging and HDD logging, power consumption by RAM logging is found to be less than the consumption of HDD logging. So it is concluded that to provide fault tolerance in extreme scale distributed computing systems, message logging protocol using RAM to store logs should be preferred over the HDD based message logging. For coordinated protocols, energy consumption patterns are similar to the patterns seen in uncoordinated protocols. The energy consumption by coordinated protocols depends upon the duration of the coordination process, which further depends upon process. Poor synchronization means a longer coordination process and more power consumption. By slowing down the fastest process, extra energy consumption can be minimized.

To make the decision about choosing suitable energy aware fault tolerance methods, an evaluation of coordinated (3 coordinate) and uncoordinated (RAM logging) fault tolerance protocols is performed using 4 NAS parallel benchmarks with 16 and 64 processes. All experiments are conducted on Lyon site of Grid5000 [28] using their energy measuring infrastructure facility. It is concluded that message logging protocols are more suitable for the applications involving less data exchange and vice versa for coordinated methods.

Longxin Zhang et al. [171] have addressed an optimization problem to maximize the reliability with energy conservation for precedence constraint tasks in heterogeneous clusters by proposing three algorithms, such as, Reliability-aware Heterogeneous Earliest Finish Time (RHFT), Reliability-aware Critical-Path-On-a-Processor (RCPOP) and Reliability Maximization with Energy Constraint (RMEC) algorithm. All the proposed algorithms have three phases: task priority establishment, processor frequency selection and task to processor mapping. In task priority establishment phase, all the tasks are prioritized according to their URank, which is a method to calculate

the topological order for directed acyclic graphs (DAG). After calculating the URanks (bottom-up approach), all the tasks are pushed in priority queue in decreasing order starting with highest priority. Once tasks are ordered, best frequency and voltage pair are chosen. This consumes less energy in executing tasks ready at the top of the queue. Along with the proposed algorithms, Hierarchical Reliability Driven Scheduling (HRDS) and Reliable Dynamic Level Scheduling (RDLS) algorithms are also used for a comparative evaluation.

To evaluate the performance of given scheduling algorithms, a large number of randomly generated DAG with different number of nodes (tasks) and real-world applications are used. For real world applications, three problems i.e. Fast-Fourier Transformation (FFT), LU decomposition and Gaussian elimination are chosen to generate task graphs. The simulation results show that in all cases, RMEC outperforms other algorithms in terms of reliability and energy consumption. Though the proposed algorithms worked well in the present scenarios, results may vary in the presence of correlated failures. So, to make the solutions more promising and applicable, consideration of models for correlated failures [166], [56] should be taken into account.

Wei Deng et al. [40] have proposed a Reliability-Aware server Consolidation stratEgy (RACE) to address a multi-objective problem with reliability and energy cost factors. A utility model that can estimate the cost of server consolidation in terms of reliability and energy efficiency whilst still mitigating SLA violations occurring due to resource demand and supply mismatch is formulated. The unified utility model is used by a genetic algorithm âĂŸimproved grouping genetic algorithm (IG2CA)âĂŹ to provide an optimized solution of the problem by choosing the best among the initial configurations provided by the proposed reliability-aware resource buffering and VM to PM mapping heuristics.

To prove the superiority of the proposed RACE server consolidation strategy, a simulation based analysis is done by using light, normal and heavy application workloads. The results of the simulation are compared with results of two other server consolidation strategies: pMapper [156] and PADD [98]. With the increase of incoming workload, the occurrence of SLA violations tend to increase due to the fluctuation in the workload and resource shortage. In the proposed method, the value of utility function is assessed before accommodating any request and performing VM

consolidation. If the value of utility function is positive, only then consolidation is considered valid. Because of the common utility function with unified SLA violation, energy costs and reliability, the proposed strategy has outperformed all other methods.

Jai-Chun Lin et al. [100] have studied the job completion reliability (JCR) and job energy consumption (JEC) for general map reduce infrastructure (GMI). The probabilistic models for worst case and best case are formulated to represent the reliability of slave nodes performing map and reduce tasks and master nodes running job tracker and name node instances. The best case corresponds to the execution of job without any interruption and worst case corresponds to the execution of job on every cold-standby node (redundant nodes) at the slave end. Along with the formulation of reliability of master and slave nodes to finish a job, energy consumption is formulated as the function of finishing time of the job. All the nodes at master end and at slave end are homogeneous and occurrence of failures is assumed following Poisson distribution. The influence of different number of cold-standby slave nodes (varies from 1 to 4 in this study) is evaluated on job completion reliability and job energy consumption. 10 jobs with different length of task execution time are considered and each of them are divided into 4096 map tasks and 1 reduce task. It is observed that increasing the number of cold-standby nodes from 1 to 2 increases the JCR but further increase in cold-standby nodes does not make any difference because of the absence of any redundancy measure at the master end. This means that in Map-Reduce framework, increasing the number of backup resources at slave nodes end does not increase reliability as long as no measure are taken at the master node end. For the best case scenario, energy consumption is less and linear with respect to map task execution time and independent from the number of cold-standby nodes such that energy consumption remains same for all the number of backup nodes. For the worst case, energy consumption is linear with respect to map-task execution time but varies according to the number of cold-standby nodes. When the best case occurs, increase in the number of cold standby nodes does not affect JEC of GMI.

After the analysis, it is concluded that General Map-reduce Infrastructure (GMI) is energy efficient but for long executing jobs, it is not reliable because of the absence of redundancy measure at the master end.

2.5.2 New Challenges and Future Research Directions

Many solutions have been proposed either to increase the reliability of the system (Table 2.1) or decrease the energy consumption of the system (Table 2.2). Some of the work done jointly in the field of reliability and energy efficiency of cloud computing systems is highlighted in the Table 2.3. Finding a solution to achieve both objectives at the same time poses new fundamental challenges, which are discussed in the following section out which, some are addressed in next chapters of this thesis.

Impact of Energy Efficiency Techniques on Reliability

Though a lot of work has done to optimize the energy management (Table 2.2) by exploiting power regulation techniques in order to make cloud computing systems energy efficient but reliability of cloud systems has left an open challenge to look at along with energy efficiency of the systems. To make the cloud computing systems energy efficient, all the energy-aware resource management techniques are usually based on the manipulation of underlying resources, which can be done either by running the resources at low-scaling mode or by turning them off. Though these methods are proved very efficient from the perspective of energy management, they have adverse effect on the reliability of the systems. Switching the resources between low scaling modes and high scaling modes using frequency and voltage scaling techniques (DVFS) causes an increase in the response time and decrease in the overall throughput of the system. This can result in a service delay and be considered as a service failure due to SLA violations. Besides this, turning servers on/off or putting resources in sleep mode more frequently makes them more failure prone than running the resources all the time. Just as the lifetime of a car brake-pads decrease with each slowdown, the reliability of server components, specifically disk drives, also decrease with each power modulation. That is why many disk manufacturers limited the start/stop power cycles of disk drives to 50,000 for their entire lifetime and also propose to keep the power cycles limited to 10 times/day to keep the overall system reliability high [175]. So the optimal solution is to make the cloud computing systems energy efficient and reliable at the same time and thus help to make the paradigm stable

and acceptable.

Impact of Failures on Energy Consumption

Many solutions are provided in the literature (Table 2.3) to evaluate the impact of utilization and energy consumption on the occurrence of failures but how much energy consumption of the system will be effected with the occurrence of failures remains unclear. It is necessary to use optimized fault tolerance methods to reduce the occurrence of failures in cloud computing systems. But to make the current fault tolerance methods more optimized in terms of energy consumption, it is important to study the relation between the failures and energy consumption. Defining this relationship will help to simultaneously increase the reliability and energy efficiency of cloud computing systems.

Multi-Objective Resource Provisioning Methods and Techniques

Most existing research is on either the reliability or energy efficiency aspect of cloud task scheduling. The resource or task scheduling can be formulated by using different optimization problems such as bin packing problem in which the available resources are assigned to the incoming tasks according to certain conditions. The resources provisioning is like a bin-making problem such that adequate number of resources need be reserved first and, after the reservation, bin-packing solutions can be used to do the optimization. In the case of under-provisioning of resources, the scheduler will not get enough resources to schedule the tasks which can lead to the service failure. On the other hand, in the case of over-provisioning, reserved resources will remain underutilized which will increase the cost of service in terms of energy consumption and other operational expenses. Rather than considering the resource and task mapping problem as a single-layer problem, it is better to consider it as a two-layer problem consisting of resource provisioning and resource scheduling. For each layer, different solutions need to be proposed to make the cloud computing systems both reliable and energy efficient.

Prediction Algorithms to Estimate both Fault Occurrence and Energy Consumption

If the occurrence of a failure or fault in the system is predictable, then important measures can be taken before the occurrence such that checkpoints can be saved with less overhead, running VMs or tasks can be migrated to more reliable physical machines. By doing this, service providers can save unnecessary wastage of power/energy that will be required to restart all the running process that were interrupted during the failure. The prediction will help to adopt the reactive and proactive failure management and energy management mechanisms, wisely. Suppose the occurrence of failure is known in advance, then the checkpointing or logging of current state of the system will start just before the occurrence of failure. Therefore, it can reduce overhead occurred due to the checkpointing of running system. If the overhead will be reduced then less number of backup resources will be required and energy consumption of the system will be reduced without compromising the reliability of the system.

Federated Clouds and their Standardization

Interconnected clouds or Federated clouds is the collection of clouds analogous to the Internet (collection of networks). Maurizio Giacobbe et al. [65] have defined the cloud federation as an ecosystem of different cloud providers that are interconnected in a cooperative decentralized computing environment. With the inter-cloud computing, reliability and energy efficiency of the cloud services will be increased by making them more dynamic and scalable. Being in the early stage, cloud computing is lacking in standardization. As the reference models and standards are available for other deployments such as the Internet, cloud deployment has not yet have any confirmable reference models and standards. As a result, most of the cloud providers have designed their own proprietary standards and interfaces. To avail the services of such clouds, applications need to get tailored according to the specific standards and interfaces. This gives rise to another problem called vendor lock-in [151]. The existence of the reference models (TCP/IP model in case of Internet) and standards for cloud computing paradigm will help the developers to implement the generic solutions following the similar attributes. The standardization will also help to regulate

the energy consumption of cloud infrastructure by making the migration of running VMs easy from one cloud vendor to another, which is yet only been done between the resources of same or different sites of the same cloud provider. With the proper set of standards or rules, the concept of inter-cloud computing will be realized more efficiently, which will make the cloud technology more reliable, affordable and eco-friendly. It is observed that the majority of time, the resources of data centers providing cloud services remains under-utilized but still the providers keep extending and upgrading their infrastructure to house the future needs for example Microsoft is adding 10000 servers per month to its data centers [112]. With the proper realization of inter-cloud computing architectures, this over spending can be avoided by sharing the resources between different cloud providers to serve the unexpected service requests in a reliable manner without violating the service level agreements.

Real Cloud Failure Traces

Although at the physical level, cloud computing services are deployed at infrastructure similar to other distributed computing systems, the working paradigm for the cloud computing systems is different from the rest of the distributed computing architectures. In most of the research literature, the empirical or statistical analysis about failures and energy consumption of the cloud computing systems is done by using traces or log files gathered from grids or clusters. For example, Peter Garraghan et al.[62] have done an empirical analysis to evaluate the effect of failures on the energy wastage of the cloud systems. The whole analysis is carried out by using traces generated by Google clusters. The occurrence of failures was deduced according to the behaviour or changes in the log data because no information is shared regarding the occurrence of these failures. Although, there are different types of failure traces present such as Failure Trace Archive [88], Google cluster traces, Computer Failure Data Repository (CFDR) for different types of distributed computing architectures such as grids, clusters and volunteer computers, there are no failure traces present for real clouds. A big gap exists in the analytical studies of cloud behaviour done by using non-cloud based traces or logs. To make the research more attractive, the cloud computing service providers

must disclose the real cloud traces for the occurred failures and energy consumption and must build public repositories or help the researchers to do so.

2.6 Summary

Although cloud computing platforms are widely used today, there are still plenty of research gaps to be addressed. Due to the large infrastructure of clouds, reliability and scalability are among the foremost concerns in cloud computing. In this chapter, various types of failures are explored that drive researchers to design the mechanisms to make the cloud computing systems highly reliable. This chapter has surveyed and critiqued a variety of methods aimed at increasing the reliability of cloud computing systems. The increase in the size and design complexity of clouds, is resulting in huge energy consumption and enormous carbon footprints. This chapter also presented a comprehensive survey of all the energy management techniques used in cloud computing systems. It is observed that the adoption of mechanisms to provide reliability in cloud computing services has impacted the energy consumption of the system. Adding backup resources, running replicated systems, storing logs provide strong fault tolerance but also increase the energy consumption. There is a critical trade-off between service reliability and energy consumption that urgently needs to be investigated. The need for a reliability-aware and an energy-aware resource provisioning policy is identified to improve the availability of cloud services whilst simultaneously reducing its energy consumption.

In the next chapter, we have proposed resource provisioning policies aiming to maximize the reliability and minimize the energy consumption of failure prone cloud computing systems while using checkpointing as fault tolerance mechanism.

Chapter 3

Reliable and Energy Efficient Resource Provisioning and Allocation in Cloud Computing

Reliability and Energy-efficiency are among the biggest trade-off challenges confronting cloud service providers. This chapter provides a mathematical model of both reliability and energy consumption in cloud computing systems and analyses their interplay. This chapter also proposes a formal method to calculate the finishing time of tasks running in a failure prone cloud computing environment using checkpointing and without checkpointing. To achieve the objective of maximizing the reliability and minimizing the energy-consumption of cloud computing systems, three resource provisioning and virtual machine (VM) allocation policies using the aforementioned mathematical models are proposed. These three policies are named Reliability Aware Best Fit Decreasing (RABFD), Energy Aware Best Fit Decreasing (EABFD), Reliability-Energy Aware Best Fit Decreasing (REABFD). A simulation based evaluation of the proposed policies is done by using real failure traces and workload models. The results of experiments demonstrated that by considering both reliability and energy factors during resource provisioning and VM allocation, the reliability and energy consumption of the system can be improved by 23% and 61%, respectively.

3.1 Introduction

Cloud computing has revolutionized the Information Technology sector by giving computing a perspective of service to users. It uses the Virtualization technology to provide a flexible and powerful computing environment as a service. Though the acceptance of cloud computing technology is increasing every day, it is still facing numerous challenges because of its complex and large-scale architecture. Reliability and Energy-efficiency are two key challenges that need careful attention and investigation. In this study, reliability is considered as the probability with which a task will finish the execution before the occurrence of a failure. A failure in the services of a cloud costs significantly to both providers and customers. A report from Ponemon institute in 2016 revealed that the average down-time cost of data centers due to outages is approximately $740,357 per year [73]. According to the Information week, each year IT outages result in the revenue loss of more than $26.5 billion [69]. The energy requirement to operate the cloud infrastructure is also increasing in proportion to the operational costs. Approximately, 45% of the total operational expenses of IBM data centers goes in electricity bills. It has been estimated that the servers mounted in Microsoft's cloud based data-centers consumes around 2 terawatts-hours (TWh) of energy per year for which the company pays approximately $2.5 billion per year as electricity bills [9]. Apart from the operational costs, huge amount of energy consumption by cloud computing infrastructure causes huge amount of carbon and green house gases emission in the environment.

To maximize the reliability of cloud computing services, all the cloud service vendors add backup resources/hosts (hosts, nodes and resources are used interchangeably in this work) and use replication as well as load balancing as fault tolerance mechanism. Adding extra resources increases energy consumption more steeply than the reliability. The key technique used to reduce the energy consumption is by running the resources on a low power scaling level or by turning off the under-utilized or idle resources such as backup resources by migrating the running virtual machines (VM) from the under-utilized resources to other resources. Turning off the backup resources will reduce the reliability of the system. For example, in the case of VM consolidation (key technique to reduce energy consumption in cloud computing systems), if a physical machine fails due to

some hardware or software issues before the completion of tasks and there are no recovery/backup resources, then all the VMs and their corresponding processes will have to start again. This will dramatically increase overheads such as energy consumption and resource utilization.

This creates a critical trade-off between the reliability and energy efficiency of the cloud computing systems. To make the cloud-computing systems reliable and energy-efficient, a mathematical framework is provided in this chapter to show the interplay of these two factors. Following are the key contributions of this chapter

1. A mathematical model to calculate the reliability and energy consumption while executing the tasks by VMs running on failure prone cloud computing resources.

2. A formal method to calculate the finishing time of the tasks executing on cloud resources in the presence of failures with and without checkpointing.

3. Resource provisioning and VM allocation policies using proposed models to optimize the reliability and energy-efficiency of the cloud computing systems.

3.2 Related Work

Most of the work in the literature either explored reliability or energy consumption of cloud computing systems. Very limited work is done by combining these two variables. This section discusses the recent works covering both reliability and energy efficiency.

It is claimed that as the operating frequency of a system increases w.r.t to supply voltage, reliability of the system increases but energy efficiency decreases [174]. This makes the task scheduling a challenge to achieve these two contradictive goals at the same time. In response to the challenge three different algorithms such as SHRHEFT, SHRCPOP and SHREERM are proposed by Longxin Zhang et al. [170]. Dynamic voltage scaling and shared recovery technique is used to regulate the energy consumption and to ensure the reliability, respectively. After performing the experiments, it is concluded that SHREERM algorithm surpasses the rest. With the same

objectives, a genetic algorithm (BOGA) for task-scheduling to optimize the reliability and energy-consumption of high performance computing systems is proposed by the same authors [169]. The performance of the proposed algorithm is compared with other two algorithms such as modified MODE and MOHEFT and the superiority of the new algorithm is shown over the other algorithms. All the approaches proposed in [170] [169] are focused specifically on high performance computing systems. It has also been assumed that at most one failure will occur during the life time of a task. In this work, this assumption is rejected with the injection of multiple failures.

Xiwei Qiu et al. [123] provided a theoretical correlation model for the fine grained measurement of reliability, power consumption and performance of cloud computing systems. A frequency scaling based power model while considering maximum CPU utilization is used where the power consumption is formulated based on variable utilization in this work while operating CPU at maximum frequency. The proposed work is analysed numerically except the reliability model which is simulated. However, in this work evaluation of the proposed formulation is done both analytically and by using simulation.

Amir Varasteh et al. [154] have studied the interplay of energy consumption and reliability while performing the VM consolidation in cloud computing systems. A fine grained mathematical model is provided to minimize the total data center cost while regulating the energy consumption and reliability. The proposed model is analysed using MATLAB based simulation where as in this work, the interplay of both metrics is shown by using simulation based results using real life data. Authors have calculated the reliability of homogeneous systems as the function of system activity (on-off cycles) where as in this work the reliability is calculated under the occurrence of failures in a heterogeneous environment. Authors have used the random VM allocation to the physical machines to test their models. However, three policies are proposed in this chapter to allocate the VMs besides the random allocation.

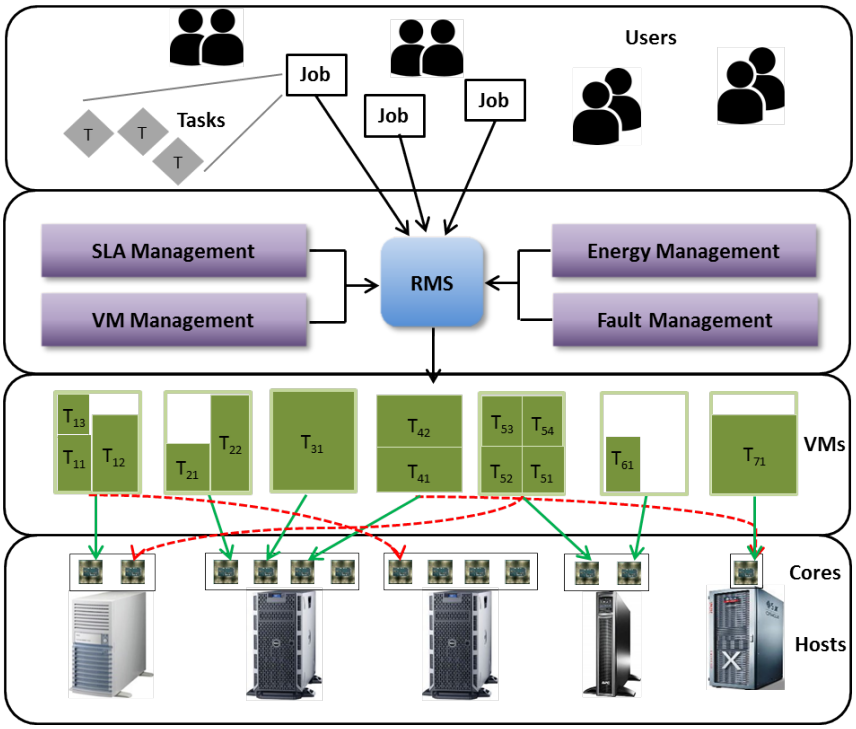


Figure 3.1: A layered System Architecture showing VMs being placed on Physical Machines (Servers) on the basis of the decisions taken at Resource Management System (RMS)

3.3 System Model

The cloud computing environment considered in this work consists of a pool P of failure prone heterogeneous resources/nodes. From the resource pool, resources get provisioned to run the heterogeneous VMs executing the tasks arrived at a specific rate. In Figure 3.1, the model is divided into four different layers. The bottom layer consisting of resources such as servers on top of which VMs are mounted. The Virtual layer is responsible for the allocation of VMs according to the decisions made at Resource Management System (RMS). The RMS is the heart of the architecture where all the reliability and energy aware resource provisioning and allocation policies are existing. The role of RMS is to gather the parameters from the energy management and fault management modules and takes the decision about the VM allocation so that the reliability of the system will be maximized and energy consumption will be minimized. Users/Brokers are submitting their tasks to the RMS seeking execution before the deadline.

On the arrival of new tasks, the current status of the resources and resource requirements of new tasks get evaluated at RMS. On the basis of the evaluation, optimized decisions about the resource provisioning and VM allocation takes place according to the proposed algorithms to regulate the reliability and energy-efficiency of the system. The number of VMs running on a provisioned node do not exceed the number of cores available on the node. One core is allocated to each VM and VMs are not allowed to share the cores with each other (such configuration can be obtained in Xen [12] hypervisor). Memory of the host is shared by the running VMs and to avoid the interference during run time, each VM has an exclusive share of the host memory. Note: VMs used in this thesis have no local or global queues where tasks will get accumulated and wait for their execution. On the arrival of a task, it gets allocated to an available VM without any delay. In case a VM is not available, new VM gets created.

3.3.1 Application Model

Cloud computing systems can be used to run a variety of applications or workloads consisting of dependent tasks such as Map-reduce applications or independent tasks such as Bag-of-Task applications. In this work, Bag-of-Task (BoT) application composed of bags or groups of independent and sequential compute intensive tasks is used because of their wide adoption in scientific and commercial organizations such as Facebook [147]. The most common examples of BoT applications are image manipulation applications (astronomy, image rendering, video surveillance), data mining applications, Monte Carlo simulations and intensive search applications.

Each BoT consists of set of independent tasks, $T = \{t_i \mid 1 \leq i \leq n\}$. Every task t_i has a corresponding length, l_i. In this work, the length of a task is measured by the number of instructions, which is the most common way to calculate the task length found in the literature [61] [14] [169]. Moreover, there are no generic methods available to calculate the task length and each method available in the literature is application specific. Each task t_i is allocated to a VM $vm_j \in VM$, where $VM = \left\{vm_j \mid 1 \leq j \leq m\right\}$ and each VM, vm_j runs a set of tasks, Γ_j, where $\Gamma_j \subseteq T$. To launch the desired number of VMs, a number of physical machines or nodes, $N = \{n_k \mid 1 \leq k \leq x\}$ get

provisioned from resource pool, P. Every task has a deadline d_i associated to it which is calculated according to the model proposed in the following section.

3.3.2 Deadline Model

Due to the occurrence of failures, fault-tolerance mechanism overheads and dynamic nature of cloud computing systems, execution delay happens, which results in SLA violations and affects the promised quality of service (QoS). In order to tolerate the impact of all the aforementioned factors while fulfilling the conditions of SLA, deadline d_i corresponding to each submitted task t_i is calculated using Equation 3.1. Defining the deadline gives the provider a time window corresponding to each task within which the task can be finished while fulfilling the SLA conditions and without any service interruption faced at the user end.

$$d_i = \begin{cases} s_i + (f.l_i), & \text{if } [s_i + (f.l_i)] < c_i \\ c_i, & \text{otherwise} \end{cases} \tag{3.1}$$

where, s_i, l_i and c_i are the starting time, task length and completion time of a task t_i, respectively [78]. f is the stringency factor that defines the deadline strictness i.e. higher the value of f is, higher deadline relaxation the task has.

3.4 Reliability Model

The given reliability is only modeled for the intrinsic failure period of a system among all three periods shown in Figure 3.2. This is because during early and wear out failure periods, supporting the system with backup/secondary resources to tolerate failures is the best solution. If a primary resource goes down, secondary resource takes charge and begins to serve. Once the primary resources are backed by secondary resources to tolerate the failures, there is nothing much to take measures about. But opting such static solutions for intrinsic failure period is very expensive as it is the longest operational and useful period during the lifetime of a system.

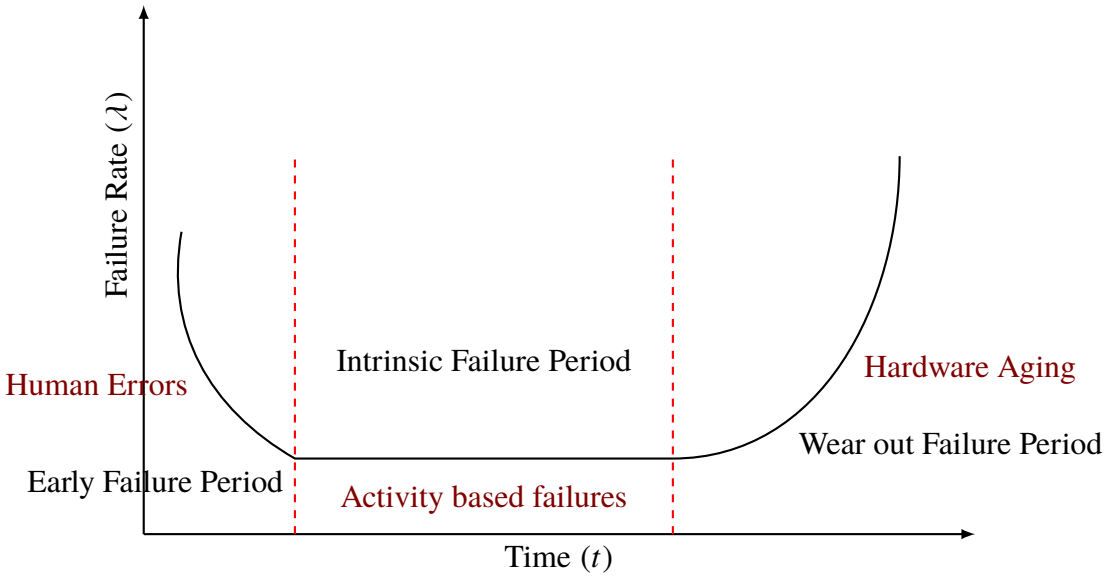
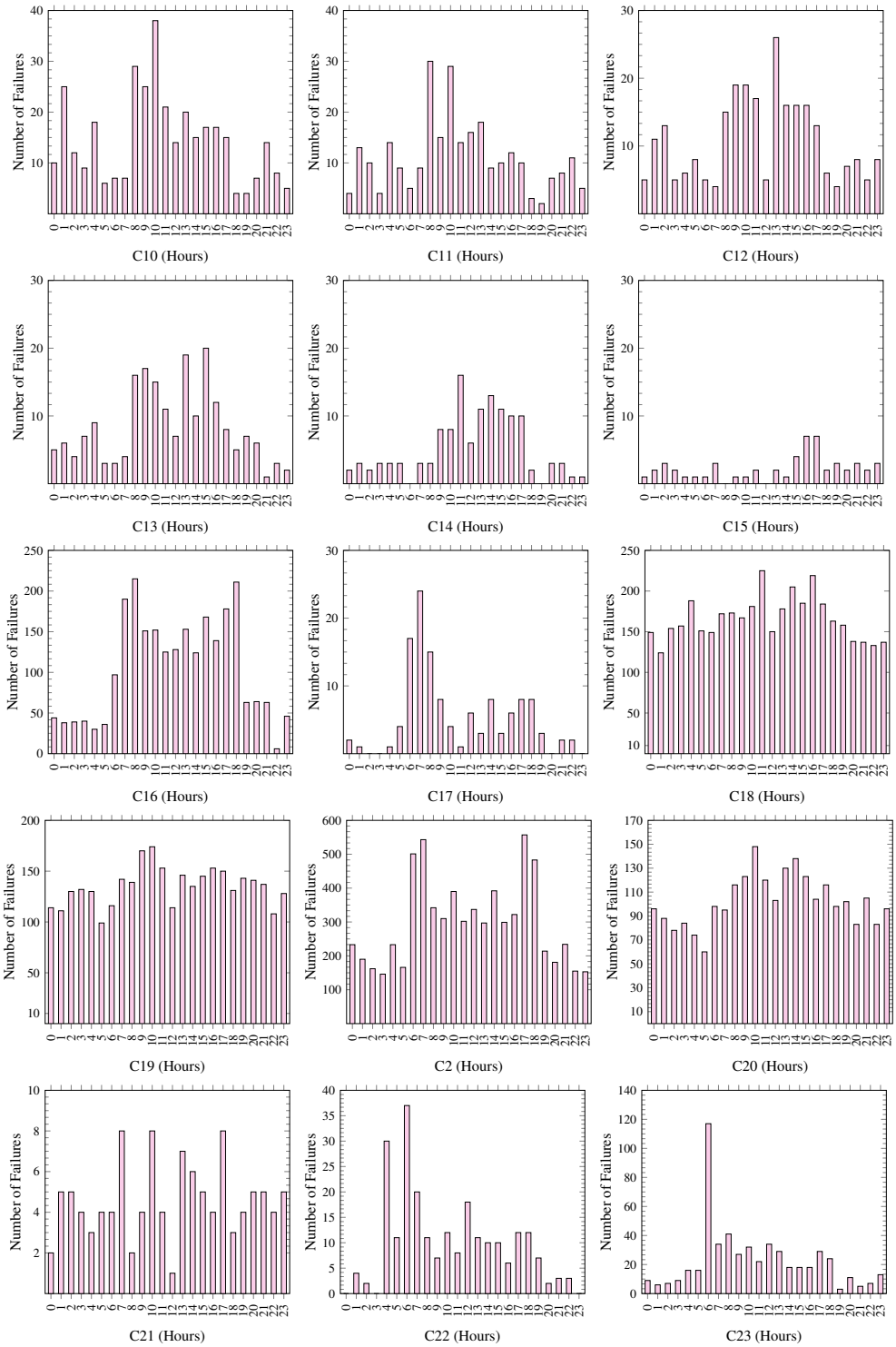


Figure 3.2: Bath Tub Curve for Reliability Modeling

During this period, the failure rate remains roughly constant and can be associated to an activity of the system such as utilization [138] and operating frequency [174]. Due to such behaviour, it is possible to make predictions with minimal error about the occurrence of failures during this period. Such estimates can be exploited further in order to reduce the number working resources which further reduce the operational expenses. Scope of such estimation provided the motivation to focus specifically on intrinsic failure period. Factors contributing to early failure period and wear out failure period such as human errors and hardware ageing, respectively can be ignored while modeling reliability during intrinsic failure period.

In order to model the reliability, we have performed statistical analysis of failure events and affected hosts by using failure information present in the failure traces provided by Los Alamos National Lab (LANL). The failure traces are downloaded from Failure Trace Archive (FTA)[88], an on-line public repository providing failure traces gathered from 26 different computing sites. The failure dataset consists of data gathered from 23 high performance computing (HPC) systems at LANL. After performing the analysis, it is observed that failure rate changes with the time. Figure 3.3 shows the failure count per hour of a day corresponding to all 23 HPC systems. For


Number of Failures
C10 (Hours)
C11 (Hours)
C12 (Hours)
C13 (Hours)
C14 (Hours)
C15 (Hours)
C16 (Hours)
C17 (Hours)
C18 (Hours)
C19 (Hours)
C2 (Hours)
C20 (Hours)
C21 (Hours)
C22 (Hours)
C23 (Hours)

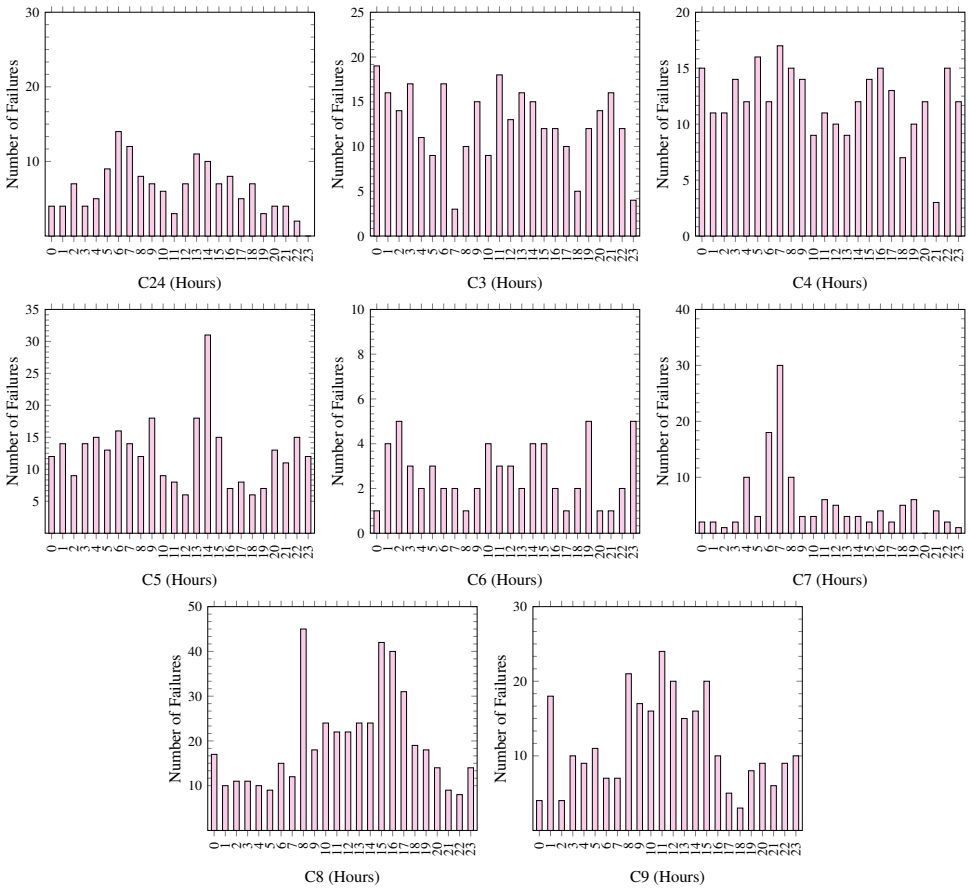


Figure 3.3: Failure Count vs. Daily Hours for Los Alamos National Laboratory (LANL) Computing Systems

most of the clusters, it is observed that during the working hours of a day, i.e., 7 am to 7 pm, the number of failure increases. This represents the correlation between the occurrence of failures and utilization/activity of the system, such that, during the working hours, the utilization/activity of the systems is higher than the non-working hours, which increases the failure rate of systems. Similar kind of impact of time on the occurrence of failures is seen in other observations as well [77], [134].

This work is mainly focused on the physical resource failures in order to bring the evaluation of reliability in accordance to other factor, such that, energy consumption, which is the hard-wired parameter of physical resources. Terms node/host/physical resources/system are used interchangeably. In this work, a VM running on a physical resource/node only failed, when the node failed.

The failure rate/hazard rate, λ_{jk} of a vm_j running on a node, n_k with utilization u_k is calculated as follows

$$\lambda_{jk} = \lambda_{max_k} \times u_k^{\beta} \tag{3.2}$$

where, λ_{jk} is the failure rate of a VM vm_j running on a node n_k with utilization u_k and λ_{max_k} is the failure rate of node n_k at maximum utilization. β (> 0) is the sensitivity factor which shows the sensitivity of the failure rate towards the utilization. When $\beta = 1$, it shows the linear relationship of λ_{jk} with u_k, which is a function of time, i.e., varying according to time [138] [134]. In this work, it is assumed that all the running VMs shares the maximum hazard rate similar to the physical node n_k which they are running on. As failures in cloud computing systems are inevitable, every node n_k in the resource pool has a Mean Time between Failures ($MTBF_{max_k}$) at maximum utilization u_{max}, which is calculated by the RMS empirically from the node failure history.

Hazard rate or Failure rate (λ_{max_k}) for a node n_k at maximum utilization is calculated as

$$\lambda_{max_k} = \frac{1}{MTBF_{max_k}} \tag{3.3}$$

The failure rate is assumed to be following Poisson distribution [174][170] and remains constant for each utilization level. So, the probability with which a VM, vm_j running on a node n_k with utilization u_k with hazard rate λ_{jk} will be able to finish the execution of all the running tasks is equal to the probability with which the longest task, l_{max_j} running on vm_j will finish the execution before the occurrence of a failure.

$$R_{vm_{jk}} = \exp^{-\lambda_{jk} \times l_{max_j}} \tag{3.4}$$

This probability is called reliability of a VM. As stated earlier, all the VMs running on a node get failed when host fails. This shows that the node and corresponding VMs share serial relationship with each other. So the reliability of a node n_k while running m VMs is calculated as the product of the reliabilities of all the VMs as follows

$$R_k = \prod_{j=1}^{m} (R_{vm_{jk}}) \tag{3.5}$$

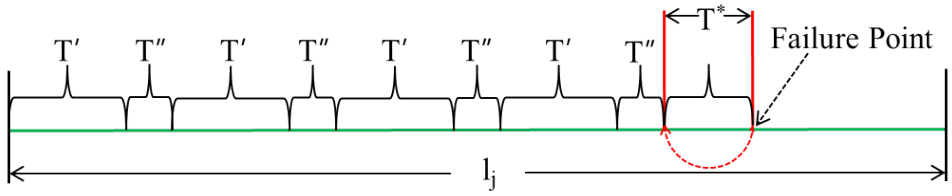


Figure 3.4: Recovery from Failure with Checkpointing

However provisioned nodes fail independently [123]. Between the provisioned nodes, neither the serial relationship nor parallel relationship exists. So the reliability of the system is calculated as the average of the reliability values possessed by all the provisioned nodes at a particular instance.

3.4.1 Fault Tolerance and Task Execution Model

In this work, two methods are used to recover from a failure i.e. Checkpointing and Restart from the beginning. In the literature, checkpointing mechanism is used intensively to provide fault-tolerance in cloud computing systems (Table 2.1). Though the solutions provided by using checkpointing method are elegant, they are very costly in terms of overheads which make it some-times impractical. If a failure will not occur, checkpointing adds the overhead of 151 hours for a job of length 100 hours in a petaflop system [119]. However, if a failure occurs, re-execution of the failed task from the last checkpoint saves a good amount of re-execution period.

Finishing Time with Checkpointing

In Figure 3.4, T' represents the checkpoint interval and T'' represents the checkpoint overhead such that time taken by the system to save the checkpoint of the running task on some stable storage. T'' for a task, t_i running on VM, vm_j is calculated as the product of maximum overhead imposed by a checkpoint, CO^{max} and VM utilization, u_j. It is assumed that the storage where all the checkpoints are getting stored is failure free. T^* represents the time required to re-execute the part of the task that was lost because of the occurrence of a failure. To further reduce the checkpointing overhead, risk based checkpointing mechanism [116] is used in this work. In risk based checkpointing, if the

expected amount of lost work before the checkpoint is smaller than the cost of checkpoint T'' then skip the checkpoint.

To calculate the length of the lost part of a failed task t_i of length l_i associated to VM vm_j that need to be re-executed on a node n_k, first it is required to calculate the interval of checkpointing T'_k for that node, which is calculated as follows by using Young's formula [167]

$$T'_k = \sqrt{2 \times T'' \times MTBF_k} \tag{3.6}$$

Suppose, $T^{\#}$ represents the part of a task that is executed before the occurrence of a failure. The number of checkpoint intervals that took place before the occurrence of a failure on a node n_k while executing the task t_i of vm_j is calculated as

$$N'_{ij} = \left\lfloor \frac{T^{\#}_{ij}}{T'_k} \right\rfloor \tag{3.7}$$

The length of the lost part of failed task t_i of vm_j that need to be re-executed is calculated as

$$T^{*}_{ij} = \left(\frac{T^{\#}_{ij}}{T'_k} - N'_{ij} \right) \times T'_k \tag{3.8}$$

Besides the checkpointing overheads and re-execution part of the failed task, time to return (TTR) from a failed state to running state also adds to the finishing time of a task. So, after the occurrence of n failures and m checkpoints, the finishing time of a task t_i of vm_j with length l_i executing on node n_k is calculated as follows

$$F_{ij} = \begin{cases} l_i + \sum_{p=0}^{n} T^{*}_{(ij)_p} + \left(T''_j \times \sum_{q=0}^{m} N'_{(ij)_q} \right) + \sum_{p=0}^{n} TTR_{(ij)_p}, & \text{if } n, m > 0 \\ l_i & \text{otherwise} \end{cases} \tag{3.9}$$

Finishing Time without Checkpointing

Due to the expensive implementation of checkpointing mechanism, restarting of the execution of a failed task or job from the beginning (Figure 3.5) is more preferable in practice because of the less overheads. The adoption of task restart mechanism is claimed on the basis of the discussions and surveys done at the Fujitsu Primergy high-performance, distributed-memory cluster named

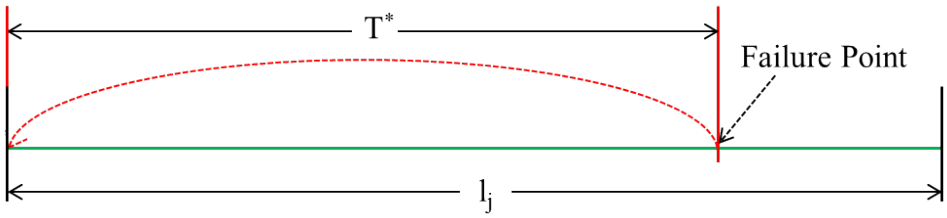


Figure 3.5: Recovery from Failure without Checkpointing

Raijin located at National Computing Infrastructure (NCI) facility in Australia[3]. As there are no checkpoints in this case, the lost part of the task that need to be re-executed, T^* is equal to the part of the task length that was executed before the occurrence of a failure. So, after the occurrence of n failures, the finishing time of a task t_i of vm_j with length l_i executing on node n_k is calculated as follows

$$F_{ij} = \begin{cases} l_i + \sum_{p=0}^{n} T^*_{(ij)_p} + \sum_{p=0}^{n} TTR_{(ij)_p}, & \text{if } n > 0 \\ l_i & \text{otherwise} \end{cases} \tag{3.10}$$

3.5 Energy Model

Many devices such as CPU, storage, memory, network interfaces and other PCI devices contribute to the power consumption of the system. But in the literature, it has been argued that CPU is the biggest power consumer despite of the advancement in the hardware and software technology [15] [84]. Based on the literature, this work is focused on the energy minimization by regulating the utilization of CPU while operating at maximum frequency. While practising resource heterogeneity, it is assumed that all the nodes in the resource pool, P have different minimum power (P_{min}) and maximum power (P_{max}) consumptions. The power consumption by a VM vm_j with utilization u_j running on a node n_k is calculated as follows

$$P_k(u_j) = (frac_k \times P_{max_k}) + ((1 - frac_k) \times P_{max_k} \times u_j) \tag{3.11}$$

[3]https://nci.org.au/systems-services/national-facility/peak-system/raijin/

$frac_k$ is the fraction of minimum, P_{min_k} and maximum, P_{max_k} power consumption for a node n_k [14]. Utilization of a VM, vm_j is calculated as the sum of normalized lengths of tasks $\in \Gamma_j$ executing on VM. The energy consumption is the amount of power consumed per unit time. In the presence of failures, the energy consumption is the sum of energy consumed while executing the task length and energy wastage, E_{waste} because of the failure overheads. So the energy consumption by a VM, vm_j running on node n_k while executing a task of length l_i in the presence of failures is calculated as follows

$$E_{vm_{ij}} = \left(P_k(u_j) \times l_i\right) + E_{waste_{ij}} \tag{3.12}$$

As shown in Equations 3.9-3.10, the finishing time of a task changes because of the occurrence of failures. Besides re-execution time, there are other factors that adds to the execution time of a task such as down-time also called Time to Return (TTR) i.e. time that system took to restart the execution after a failure. During the down-time, system is in non-working condition so it does not contribute to the energy wastage.

3.5.1 Energy wastage with checkpointing

For the checkpointing, the energy wastage is further splitted into the energy consumed while saving the checkpoints, T'' and energy consumed while re-executing the lost part of a task, T^*.

$$E_{waste_{ij}} = E_{T''_{ij}} + E^*_{ij} \tag{3.13}$$

The power consumption while saving the checkpoints on a disk, P_{chkpt} consumes less power than power consumption during the execution of a task. This is because during the creation of checkpoints the activity of CPU decreases (biggest energy consumer) and activity of I/O controllers i.e. Direct Memory Access (DMA) controller increases in order to perform read/write operations on backup drives. So, the energy wastage for task t_i running on a VM vm_j while using checkpointing is calculated as follows

$$E_{T''_{ij}} = \begin{cases} f_{chkpt} \times P_{min_k} \times \left(T''_j \times \sum_{q=0}^{m} N'_{(ij)_q}\right), & \text{if } m > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.14}$$

The energy consumed while re-executing the lost part of the task t_i because of the occurrence of n failures is calculated as

$$E^*_{ij} = \begin{cases} P_k(u_j) \times \sum_{p=0}^{n} T^*_{(ij)_p}, & \text{if } n > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

In the absence of checkpointing, the only energy that is wasted is the energy consumption while performing the re-execution of the lost part of a task because of the occurrence of n failures and is calculated using Equation 3.15. So, the total energy consumption by x provisioned nodes allocated to m VMs while finishing all the tasks of BoT application in the presence of failures using checkpointing and without checkpointing is calculated as follows

$$E_{total} = \sum_{k=1}^{x} \sum_{j=1}^{m} E_{vm_{jk}} \tag{3.16}$$

3.6 Resource Provisioning and VM Allocation Policies

With the given BoT application consisting of a set of BoTs with n independent tasks in each and a pool of failure prone cloud resources, the challenge is how to provision the resources and allocate the VMs executing the tasks in order to maximize the reliability and minimize the energy consumption while keeping the turnaround time of every task less than corresponding deadline.

Before provisioning the physical resources from the pool of resources, P, the first challenge was to calculate the number of VMs need to be instantiated to execute the tasks. Problem of allocation of tasks to VMs is formulated as a bin-packing problem [133] where the VM is considered as a bin and capacity of a bin is the utilization level of VM. Each task, t_i of a BoT B had a corresponding utilization, which is calculated by normalizing the task length, l_i w.r.t to the task of maximum length l_{max} in B. The minimum number of VMs, N_{min} required to instantiate to allocate all the n tasks of B is calculated as follows

$$N_{min} = \left\lceil \sum_{i=1}^{n} \frac{l_i}{l_{max}} \right\rceil \tag{3.17}$$

Before instantiating new VMs, VM provisioning and task allocation algorithm (Algorithm 1) checks the available capacity on the running VMs and tried to allocate maximum number of tasks to the

Table 3.1: Nomenclature used in algorithms and functions

Notation	Explanation
B	Set of Bag of Tasks
T	Set of Tasks in a Bag
t_i	i_{th} task
l_i	Length of i_{th} task
vm_j	j_{th} virtual machine
l_{max_j}	Length of a longest task in T corresponding to vm_j
u_j	Utilization corresponding to j_{th} VM
R	List of Resources/Nodes in P
R_{sorted}	Sorted list of Resources
$\mathfrak{R}$	List of Provisioned Resources
V	Set of Virtual Machines (VMs)
V_{sorted}	Sorted list of VMs
n_k	k_{th} node
λ_k	Current Hazard Rate of k_{th} node
P_k	Current Power consumption of k_{th} node
$MTBF_k$	Current Mean Time Between Failure of k_{th} node
Ψ_k	Ratio of $MTBF_k$ and P_k
RC_k	Remaining cores of k_{th} node
rel_{jk}	Reliability of j_{th} virtual machine on k_{th} node
pow_{jk}	Power consumption of j_{th} virtual machine on k_{th} node

running VMs (step 6-16). For the remaining tasks, minimum number of new VMs need to be instantiated is calculated (step 19-23). After calculating the minimum number of VMs, optimized allocation of remaining tasks is done using Best Fit Bin Packing algorithm [133] while taking the status of all the running VMs into consideration.

During the creation of new VMs to allocate the tasks, physical machines from the pool of heterogeneous resources, P are provisioned and allocated to the VMs (Algorithm 2). The physical nodes in P (Figure 3.1) have different hazard rate and power profiles which are recorded by RMS during previous runs. The resources are chosen on the basis of the objective of the cloud provider to maximize the reliability or minimize the energy consumption or achieve both objectives at the same time. Depending on the objectives of the cloud provider three list based greedy policies such as Reliability Aware Best Fit Decreasing (RABFD), Energy Aware Best Fit Decreasing (EABFD)

Algorithm 1: VM Provisioning and Task Allocation

Input: *Bag of Tasks*, **B** *and List of VMs*, **V**
Output: *Set of Provisioned VMs and Allocated Tasks*

```
1  //Calculating the normalized length of each task in B;
2  for i ∈ B do
3      n_i = l_i/l_max;
4      N ← n_i;
5  //Allocating tasks to currently running VMs;
6  for i ∈ B do
7      for j ∈ V do
8          if (n_i == 1.0) then
9              break ;
10         u_j ← vm_j.currentUtilization();
11         tempUtl = u_j + n_i ;
12         if (tempUtl ≤ 1.0) then
13             t_i ← vm_j.allocateVM(B);
14             u_j = u_j + n_i;
15             vm_j ← u_j.setUtilization();
16             break;
17 //For unallocated tasks, new VMs gets instiantiated;
18 B ← B.unAllocatedTasks();
19 //Calculate the minimum no. of VMs using Equation 3.17;
20 N_min ← getVMCount();
21 //Provision physical resources and allocate new VMs (Algorithm 2);
22 for k ∈ N_min do
23     vm_k ← AllocatePhysicalResources();
24 //Allocate unallocated tasks to new VMs;
25 Goto 6
```

and Reliability-Energy Aware Best Fit Decreasing (REABFD) are proposed to rank the resources before their provisioning and allocation to VMs. As the base line policy, Opportunistic Load Balancing (OLB)[20] is used. While allocating VMs to physical machines, RMS first checks the failure status and availability of idle cores on the provisioned and active physical machines $\in \mathfrak{R}$ and allocate VMs to the idle cores, if available (step 11-21). If the resource requirement for a VM, vm_j is not fulfilled by physical machines $\in \mathfrak{R}$, then a new physical machine is provisioned from P (step 23).

3.6.1 Reliability Aware Best Fit Decreasing (RABFD)

In this policy (Function 1), all the VMs executing tasks get sorted in decreasing order according to their utilization and all the physical resources get sorted in increasing order according to their hazard-rate corresponding to the maximum utilization (Equation 3.3). After sorting, the resources are provisioned and VMs are allocated in such a way that the VM with maximum utilization get allocated to a physical resource with minimum hazard-rate.

3.6.2 Energy Aware Best Fit Decreasing (EABFD)

This policy is designed to optimize the energy consumption by VMs (Function 2). In this policy, all the physical resources get sorted in the increasing order according to their power consumption corresponding to the current utilization (Equation 3.11), so that the VM with maximum utilization get allocated to a physical resource with minimum power consumption (Algorithm 2).

3.6.3 Reliability-Energy Aware Best Fit Decreasing (REABFD)

The objective of this policy is to optimize the reliability and energy consumption both at the same time. In the given policy (Function 3), the ratio of $MTBF_k$ and power consumption, P_k corresponding to the current utilization for each physical resource is used to rank the resources. All the resources get sorted in decreasing order according to the calculated ratio. A VM with maximum utilization get allocated to a node with highest ratio value.

Algorithm 2: Resource Provisioning and VM Allocation

Input: *List of Resources*, **R** *and List of VMs*, **V**
Output: *Set of Provisioned Resources and Allocated VMs*

```
1  if (Policy == RABFD) then
2      R_sorted ←ReliabilityAware(R);
3  if (Policy == EABFD) then
4      R_sorted ←EnergyAware(R);
5  if (Policy == REABFD) then
6      R_sorted ←ReliabilityAndEnergyAware(R);
7  else
8      R_sorted ←OpportunisticLoadBalancing(R);
9  for j ∈ V do
10     VM_cores_j ← vm_j.coresRequired();
11     for k ∈ ℜ do
12         if ((ℜ_k ≠ failed) && (RC_k ≥ VM_cores_j)) then
13             r_k ← vm_j.allocateHost();
14             RC_k = RC_k − VM_cores_j;
15             //Calculate VM reliability by using Equation 3.4;
16             rel_jk ← vm_j.calculateReliability();
17             //Estimate VM power consumption by using Equation 3.11;
18             pow_jk ← vm_j.estimatePower();
19             if (RC_k == 0) then
20                 R_sorted = R_sorted − ℜ_k;
21             break;
22 //In case of unallocated VMs, New host is provisioned from resource pool, P;
23 if (vm_j.unallocated() == true) then
24     ℜ ← R_sorted;
25     Goto 11
```

Function 1: Reliability Aware Best Fit Decreasing(REABFD)

```
1  function ReliabilityAware(R)
2  for k ∈ R do
3      λ_max_k ← r_k.calculateHazardRate();
4  for  k ∈ R do
5      R_sorted ← λ_max_k.sortHazardRateIncreasing();
6  return R_sorted;
```

Function 2: Energy Aware Best Fit Decreasing(REABFD)

```
1 function EnergyAware(R)
2 for k ∈ R do
3     P_k ← r_k.calculatePowerConsumption();
4 for k ∈ R do
5     R_sorted ← P_k.sortPowerIncreasing();
6 return R_sorted;
```

Function 3: Reliability and Energy Aware Best Fit Decreasing(REABFD)

```
1 function ReliabilityAndEnergyAware(R)
2 for k ∈ R do
3     MTBF_k ← r_k.calculateMTBF();
4     P_k ← r_k.calculatePowerConsumption();
5     Ψ_k ← (MTBF_k)/(P_k);
6 for k ∈ R do
7     R_sorted ← Ψ_k.sortMTBFPowerRatioIncreasing();
8 return R_sorted;
```

3.6.4 Opportunistic Load Balancing(OLB)

This policy is used as a baseline policy. In OLB, no criteria is used to rank the physical resources and no preprocessing of VMs is done based on their utilization as done in the previous policies. All the VMs executing tasks get allocated in random order as they are arriving to the next available resource.

3.7 Performance Evaluation

After doing a Matlab based verification of all the proposed mathematical models and algorithms, a simulation based validation of the same is done using a realistic cloud computing architecture purposed in Figure 3.1 and using configuration parameters given in Table 3.2. In order to simulate the architecture, a well known cloud computing simulator $'CloudSim'$ [27] is extended by adding failure injectors and fault tolerance mechanisms.

Table 3.2: Simulation Configuration Parameters

Input Parameters	Values
Stringency Factor (f)	1.3
Sensitivity Factor (β)	1
Maximum Checkpointing Overhead (CO^{max})	20 secs
Idle Power Checkpointing Fraction (f_{chkpt})	1.15

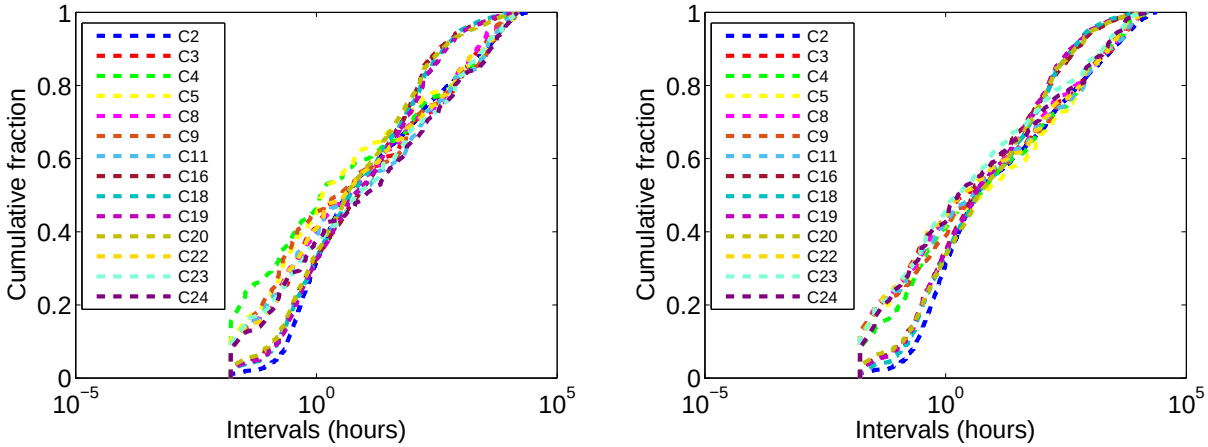


Figure 3.6: CDF of (a) Time between Failures (TBF) (b) Time to Return (TTR)

3.7.1 Simulation Setup

The hardware configuration of more than 2000 hosts of the data center is taken from Los Alamos National Laboratory (LANL) data set of Failure Trace Archive (FTA)[88]. FTA is an online public repository providing failure traces gathered from 26 different computing sites. This work has used LANL traces specifically because of the precise details provided regarding the failure start time and end time, causes of failures and node configuration. Traces from LANL systems were collected between year June 1996 to November 2005 and covers data from 23 high performance computing systems consisting of 4750 nodes in total. The mean time between failures (MTBF) and mean time to return (MTTR) for each node at maximum utilization is calculated by using the failure information provided in the traces. To calculate an accordant value of MTBF and MTTR,

Table 3.3: Workload Generation Parameters

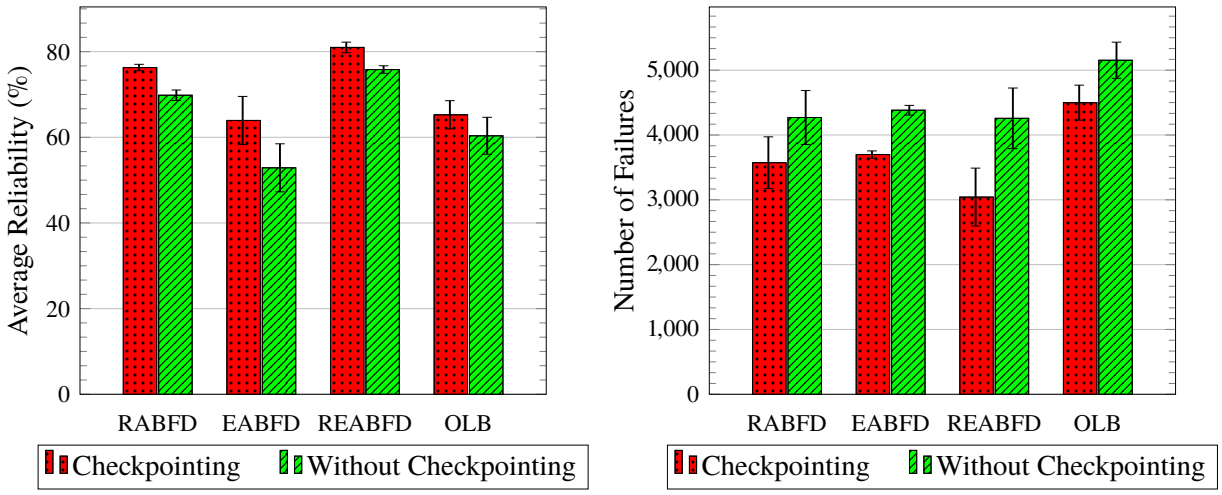
Input Parameters	Distribution	Values
Inter-arrival time (BoT)	Weibull	Scale = 4.25, Shape = 7.86
Number of Tasks per BoT	Weibull	Scale = 1.76, Shape = 2.11
Average runtime per task	Normal	Mean = 2.73, SD = 6.1

only the nodes with event count more than 3 in the traces are considered. In order to have a clear representation of cumulative distribution functions (CDFs), 14 clusters with more than 120 failure events are chosen. Figure 3.6 shows the CDFs of time between failures (TBF) i.e. availability events and time to return (TTR) i.e. unavailability events corresponding to all the 14 clusters. From CDFs, same behaviour can been seen between the occurrence of availability and unavailability events. Parameter fitting tests are performed for various distributions and found Gamma, Weibull and log-normal distributions as good fits for both availability and unavailability events, which varies system to system.

To calculate the power consumption, the values of minimum and maximum power consumption corresponding to a node are taken from spec2008 benchmark[4]. To select the realistic data center nodes, the core count and memory capacity of the nodes is matched with the values provided in the traces. This approach is used by Peter Garraghan et al. [62] by using Google trace logs. On the basis of the match, Intel Platform SE7520AF2 Server Board, HP ProLiant DL380 G5, HP ProLiant DL758 G5, HP ProLiant DL560 Gen9 and Dell PowerEdge R830 are selected as 2, 4, 32, 128 and 256 core nodes with 4GB, 16GB, 32GB, 64GB and 256GB memory, respectively. To keep the deadlines corresponding to tasks moderate [78] and to keep a strict linear relationship [174] between the utilization and reliability, values for stringency factor, f and sensitivity factor, β are set to 1.3 and 1, respectively. Idle power checkpointing fraction, f_{chkpt} and Maximum checkpointing overhead, CO^{max} is set to 1.15 and 20 secs, respectively [47].

To generate the BoTs workload, model proposed by Iosup et al. [75] is used with parameters given in Table 3.3. In the analysis, it is established that the arrival of jobs behave differently in peak

[4]https://www.spec.org/power_ssj2008/results/



(a) Average Reliability

(b) Number of Failures

Figure 3.7: Results for Reliability Evaluation

and off-peak times. To provision the enough number of nodes for the fair evaluation of proposed policies, the inter-arrival time is modeled using peak time workload following Weibull distribution with scale and shape parameters equal to 4.25 and 7.86, respectively. Every incoming BoT consists of 2^x tasks where x follows Weibull distribution with scale and shape parameters equal to 1.76 and 2.11, respectively. The length or execution time of each task in a BoT is modeled as normal distribution with mean and standard deviation (SD) values equal to 2.73 and 6.1, respectively.

3.7.2 Results and Discussions

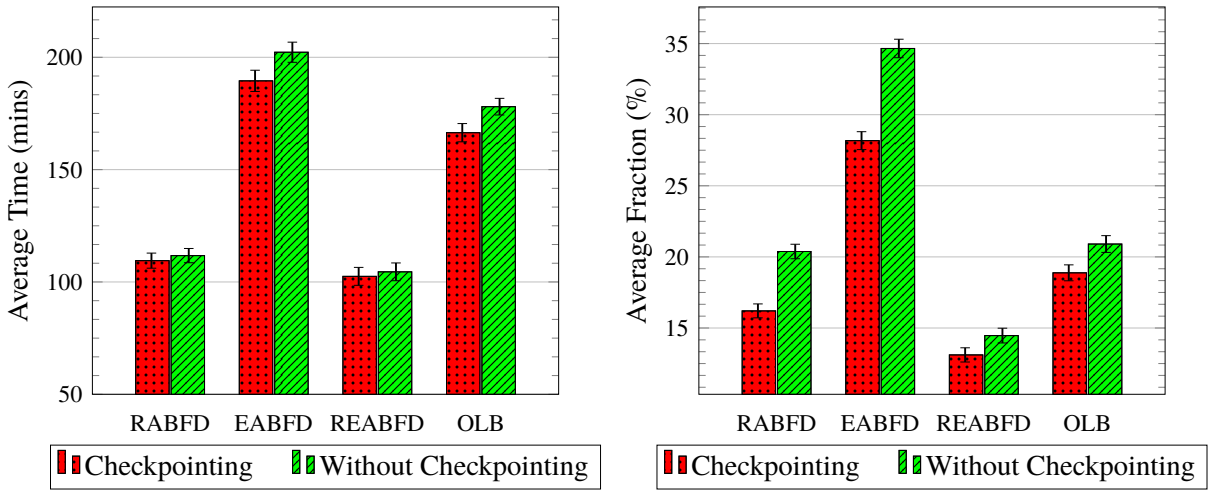
Performance evaluation of the proposed resource provisioning and VM allocation policies with and without fault tolerance mechanisms is done in terms of reliability, finishing time and energy-efficiency. All the simulations are performed by using 1000 BoTs with total number of tasks ranges between 100000 to 120000. All the reported results are the average of 50 simulations with 95% confidence interval.

Evaluation of Reliability

Figure 3.7a presents the average reliability for each policy using checkpointing and without checkpointing. It can be seen that system using REABFD policy possessed better reliability by approximately 5% from RABFD, 16% from OLB and 17% from EABFD with checkpointing besides lowest failure count (Figure 3.7b). Also in the scenario of without checkpointing, REABFD gave better reliability by 6% from RABFD, 15% from OLB and 23% from EABFD. Figure also shows that policies with checkpointing provided better reliability by 5% to 9% than without checkpointing while being affected by less number of failures (Figure 3.7b). This is due to the fact that after an event of a failure, the task length generally gets reduced in the process of recovery from the last checkpoint (if any). For shorter task length, the system possess high reliability (Figure 3.4) and has higher probability to execute the task without or before the occurrence of a failure. However, in the scenario without checkpointing, the task restarts from the beginning after an event of a failure. As the size of a resubmitted task remains same, reliability of a system also remains unchanged and relatively higher than checkpointing scenario. Moreover, in terms of reliability, REABFD without checkpointing has achieved almost similar reliability achieved by RABFD with checkpointing.

Evaluation of Execution Time

Figure 3.8a shows the average turnaround time, which is the time taken by each task of BoT application to finish. It can be seen that system using REABFD policy to provision the resources has the minimum turnaround time such that it took less time than what other policies took to finish the same application. This is because of the less overheads incurred because of lowest failure count (Figure 3.7b). REABFD policy has achieved better turnaround time by 7% from RABFD, 39% from OLB and 46% from EABFD for both checkpointing and without checkpointing scenarios. However, all the proposed policies have achieved better turnaround time while using checkpointing by 7% than without checkpointing because of less re-execution overheads occurred during the event of a failure. The achieved improvement in the turnaround time further justifies the improvements achieved in reliability (Figure 3.7a) and energy consumption (Figure 3.9a) for the REABFD policy.



(a) Average Turnaround Time

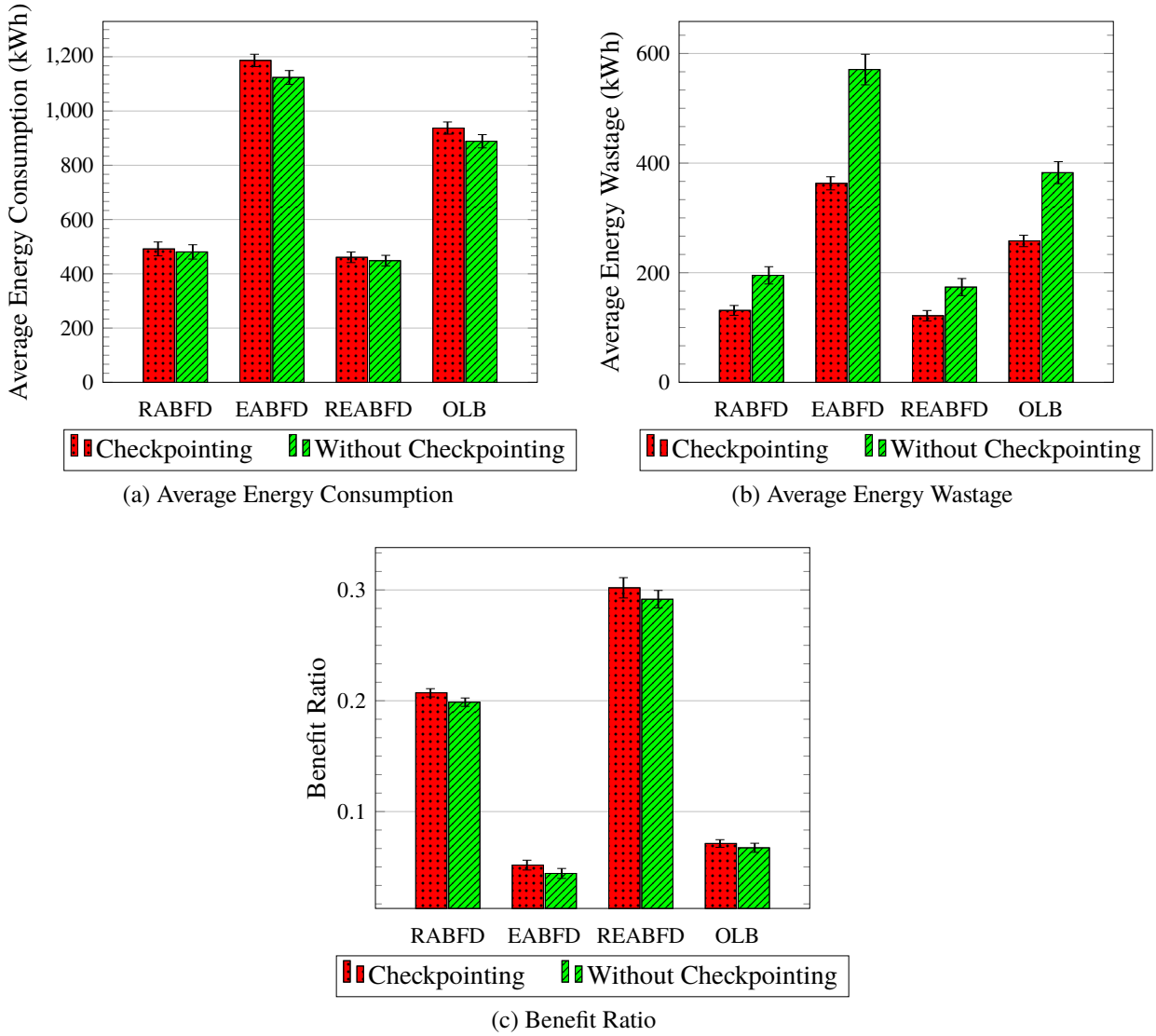
(b) Average Deadline-Turnaround Time Fraction

Figure 3.8: Results for Execution Time Evaluation

Figure 3.8b shows, while executing a task by what percentage the makespan is exceeded from the predefined deadline calculated by using Equation 3.1. It can be seen clearly that for the scenarios without checkpointing, the makespan is exceeded more up to 7% in comparison to the scenarios with checkpointing. This is because if a failure hits a scenario without checkpointing, then the re-execution overhead is huge which is found to be approximately 36% higher in comparison to checkpointing. This makes the deadlines violated with greater margin than the checkpointing scenarios. Among all the proposed policies, REABFD again outperforms the other proposed policies by exceeding less by 3%, 6% and 15% with checkpointing and 6%, 7% and 20% without checkpointing in comparison to RABFD, OLB and EABFD, respectively.

Evaluation of Energy Consumption

Figure 3.9a shows the average energy consumption incurred by all the policies with and without checkpointing. The energy consumption by REABFD is less in comparison to other policies with minimum difference of 7% from RABFD with and without checkpointing and maximum difference of 61% from EABFD with and without checkpointing. An interesting behaviour is seen for EABFD policy as it consumed the maximum energy despite of the fact that it focuses on the provisioning



(a) Average Energy Consumption

(b) Average Energy Wastage

(c) Benefit Ratio

Figure 3.9: Results for Energy Efficiency Evaluation

of most energy efficient resources. From the given behaviour, it can be argued that the results are adverse. Rather than reducing the energy consumption, the system ends up consuming more energy due to the energy losses incurred because of failure overheads. In fact, it is better to use the random resource provisioning (OLB policy) than the energy aware resource provisioning in the presence of failures. In terms of energy wastage (Figure 3.9b), again REABFD outperforms all the policies with minimum improvement of approximately 8% and 11% over RABFD policy with and without checkpointing, respectively and maximum improvement of 67% and 70% over EABFD policy with and without checkpointing, respectively. Absence of any fault tolerance mechanism such as checkpointing further added to these energy losses up to 36% because of the large re-execution overheads.

To measure the effectiveness of all the proposed policies in terms of reliability and energy consumption at the same time, a benefit ratio (Figure 3.9c) is used, which is the ratio of reliability and energy consumption. It is inferred that the policies considering reliability factor (RABFD and REABFD) while provisioning the resources have the better benefit ratio than the policy considering only energy-efficiency (EABFD) and policy considering neither reliability nor energy-efficiency (OLB). Among all the policies, REABFD performed better by improving the benefit ratio by 29% over RABFD, 82% over EABFD and 76% over OLB with checkpointing. However, in the scenario without checkpointing REABFD policy gave better value of benefit ratio by 34% over RABFD, 85% over EABFD and 78% over OLB.

3.8 Summary

In this chapter, the problem of reliability and energy aware resource provisioning in cloud computing systems is addressed. In solving this problem, a scalable and elastic cloud computing architecture is proposed with three list based greedy algorithms such as Reliability Aware Best Fit Decreasing (RABFD), Energy Aware Best Fit Decreasing (EABFD) and Reliability-Energy Aware Best Fit Decreasing (REABFD). For fault tolerance, both re-execution from the beginning and checkpointing mechanism for task recovery are considered. After performing extensive experiments, following

conclusions are drawn

1. If the emphasis is given only to the energy optimization without considering reliability in a failure prone cloud computing environment, then the results will be contrary to the expectations. Rather than reducing the energy consumption, the system ends up consuming more energy due to the energy losses incurred because of failure overheads.

2. Among the proposed policies, Reliability-Energy Aware Best Fit Decreasing (REABFD) policy outperformed all the other policies and revealed that if both reliability and energy efficiency factors of resources are considered at the same time then both factors can be improved to a larger extent than being regulated individually.

This chapter provided static reliability and energy aware resource provisioning and VM allocation policies. However, the elastic nature of cloud computing environment changes the number of initially provisioned resources dynamically. Therefore, in the next chapter we have proposed a failure-aware VM consolidation mechanism which dynamically increases and decreases the provisioned physical resources while considering their failure characteristics.

Chapter 4

Failure-aware Energy-efficient VM Consolidation in Cloud Computing Systems

VM consolidation is an important technique used in cloud computing systems to improve energy efficiency. It migrates the running VMs from under utilized physical resources to other resources in order to reduce the energy consumption. But in a cloud computing environment with failure prone resources, focusing solely on energy efficiency has adverse effects. If the reliability factor of resources is ignored then the running VMs may get consolidated to unreliable physical resources. This will cause more failures and recreations of VMs, thus increasing the energy consumption. To solve this problem, this chapter proposes a failure-aware VM consolidation mechanism, which takes the occurrence of failures and the hazard rate of physical resources into consideration before performing VM consolidation. A failure prediction technique based on exponential smoothing is proposed to trigger two fault tolerance mechanisms (VM migration and VM checkpointing). A simulation based evaluation of the proposed VM consolidation mechanism is conducted by using real failure traces. The results demonstrate that by using the combination of checkpointing and VM migration with the proposed failure-aware VM consolidation mechanism, the energy consumption of cloud computing system is reduced by 34% and reliability is improved by 12% while decreasing the occurrence of failures by 14%.

4.1 Introduction

Cloud computing has emerged as a breakthrough computing paradigm. Since its emergence, almost every sector from business organizations to educational institutions has embraced it. In a study from more than 1000 organizations ranging from SMBs to large enterprises published by RightScale in 2017 [128], 80% of them are already using cloud services and 14% are planning to adopt it. Users of cloud computing access the service using easy-to-use portals such as AWS management console without knowing about the underlying system. To provide such an abstract view of the system, cloud computing systems have to perform many complex operations besides managing a large underlying infrastructure. Due to such complex operations, cloud computing systems confront service providers with many challenges such as security, sustainability, reliability, management etc. [155]. Among all the challenges, reliability and energy consumption are the two key challenges investigated in this work. In this study, reliability is termed as the probability with which a task will finish the execution before the occurrence of a failure. In cloud computing systems failures can occur at two levels, resource failures and service failures. Resource failures occur by the outage of physical computing resources on which the cloud computing services are mounted. However, service failures occurs if users are not able to access the cloud computing service or cloud provider is not able to provide the service promised in Service Level Agreement (SLA). Resource failure can lead to a service failure but service can fail in the presence of working resources during peak loads [126]. But from the recent outages in AWS [7], Netflix[125], Twitter [85], Facebook and Instagram [124], it can be concluded that resource failures are the biggest reason behind the service outage.

Failures in physical resources are inevitable due to hardware failure or software failure. It has been argued that a system with 100,000 processors experiences a failure every couple of minutes [50]. If adequate failure management measures are not taken then resource failure can lead to service failure which costs significantly to both users and provider. A report released by Ponemon institute in 2016 revealed that the average down-time cost of data centers due to outages is approximately $740,357 per year [73]. To tolerate the occurrence of resource failures in order to avoid the

occurrence of service failures and to increase the service reliability, cloud computing providers have adopted many fault tolerance techniques such as resource redundancy and replication and load balancing [137]. But improving the reliability of cloud computing services using redundant resources increases the energy consumption severely, which is already an existing challenge. It has been reported that servers mounted in Microsoft's cloud based data centers consumes approximately 2 terawatt-hours (TWh) of energy per year for which the company pays approximately $2.5 billion per year as electricity bills [9]. Most of the servers in such data centers are sitting idle and are deployed to accommodate peak load in order to avoid an outage [158].

The energy consumption can be saved by reducing the number of active but idle or underutilized resources, which on the contrary reduces the reliability of the system. This creates a critical trade-off between these two metrics. In this work, a failure-aware VM consolidation policy is proposed to reduce the energy consumption dynamically without compromising the reliability of the system. The proposed method is incorporated in a cloud computing framework using a failure and energy aware resource provisioning and VM allocation policy [138]. To provide fault tolerance both reactive (checkpointing) and proactive (VM migration) mechanisms are used. In order to reduce the overheads of the adopted fault tolerance mechanisms, failure prediction is introduced. Following are the contributions of this work

1. Mathematical models to calculate the reliability, energy consumption and finishing time while taking failures, VM migration and VM checkpointing into account.

2. Resource and VM management policies based on the proposed mathematical models to optimize the reliability and energy-efficiency with the integration of failure prediction, VM migration and checkpointing.

3. Failure-aware Energy-efficient VM consolidation policy to reduce the energy wastage while considering reliability factors of the physical machines.

4.2 Related Work

Reliability and energy efficiency of cloud computing systems are the important objectives that the research community is focusing on. This section covers the works that are recently done in reliability and energy consumption jointly in cloud computing systems.

Ao Zhou et al. [173] have targeted the data transfer delay and network resource consumption problem while recovering from failures using K-fault tolerance replication mechanism. A three step solution consisting of resource allocation, VM placement and VM recovery is provided. For VM recovery from failures, solutions are provided to minimize the data loss and reduce processing delays while taking VM proximity into consideration. With the same objective of maximizing the reliability and minimizing the execution delays while using replication for fault tolerance, Guoqi Xie et al. [164] have proposed four heuristic algorithms to optimize the computing cost along with the aforementioned metrics. Unlike [173], authors have used complex workflow applications to evaluate the proposed algorithms. Though the proposed methods provide high fault tolerance using replication but the costs for resource usage and energy consumption are ignored. However, we have used other fault tolerance mechanisms such as checkpointing and VM migration because of high usage cost of replication while ensuring high reliability and minimizing energy consumption. We have also explored heterogeneity and dynamic nature of cloud computing systems by employing different hardware types and VM consolidation and also used real time failure traces to inject failures in the simulated cloud computing environment.

While focusing on reliability, computing energy and cooling energy, X. Li et al. [97] have proposed two line-based scheduling algorithms i.e. Ella-W and Ella-B. Mathematical models to rank the physical machines in terms of reliability and energy consumption are also provided and metrics combining all the factors are proposed to evaluate the algorithms. Similar to our work, authors have done a simulation based study of the proposed methods using real failure and workload traces. In contrast to the proposed work, we are using checkpointing and VM migration to tolerate failures besides the failure and energy aware measures being taken during the resource provisioning and allocation phase. We have also used time series analysis to predict the occurrence of failures

and also exploit the dynamic nature of cloud computing systems by employing reliability aware VM consolidation to regulate the energy consumption dynamically.

4.3 System Architecture

The targeted cloud computing environment in this study is provided in chapter 3. In the given architecture, Resource Management System (RMS) is the main focus of this work. All the decisions about allocation and migration of VMs are taken by RMS where all the failure and energy aware resource provisioning, allocation and VM migration policies are incorporated. Users/Brokers are submitting their tasks to the RMS seeking execution before the deadline. In this work, Bag-of-Task (BoT) application is used because of their wide adoption in scientific and commercial organizations such as Facebook [147]. Deadline d_i associated to a task, t_i is calculated as $(s_i + (f \times l_i))$, where s_i and l_i are the starting time and task length, respectively (Equation 3.1). f is the stringency factor that defines the deadline strictness i.e. higher the value of f is, higher deadline relaxation the task has. Rather than rejecting a task for a deadline miss (hard deadline), the soft deadline concept is adopted which means it reduces the value of the computation for the users [26]. More the execution of a task is delayed, more the value is reduced. In case of missing a deadline, the remaining value ϑ_i of a task t_i is calculated as follows

$$\vartheta_i = \begin{cases} 1 - \left(\frac{c_i - d_i}{d_i}\right), & \text{if } c_i > d_i \\ 1, & \text{otherwise} \end{cases} \tag{4.1}$$

4.3.1 Reliability Model

In order to model reliability of a system, a statistical analysis of failure information acquired from the failure traces provided by LANL is performed in Chapter 3. While following the same approach and to make the analysis more diverse, a similar kind the statistical analysis of failures is performed in this chapter by using failure information present in the Grid5000 failure traces. The failure traces are downloaded from Failure Trace Archive (FTA)[88], an online public repository providing failure

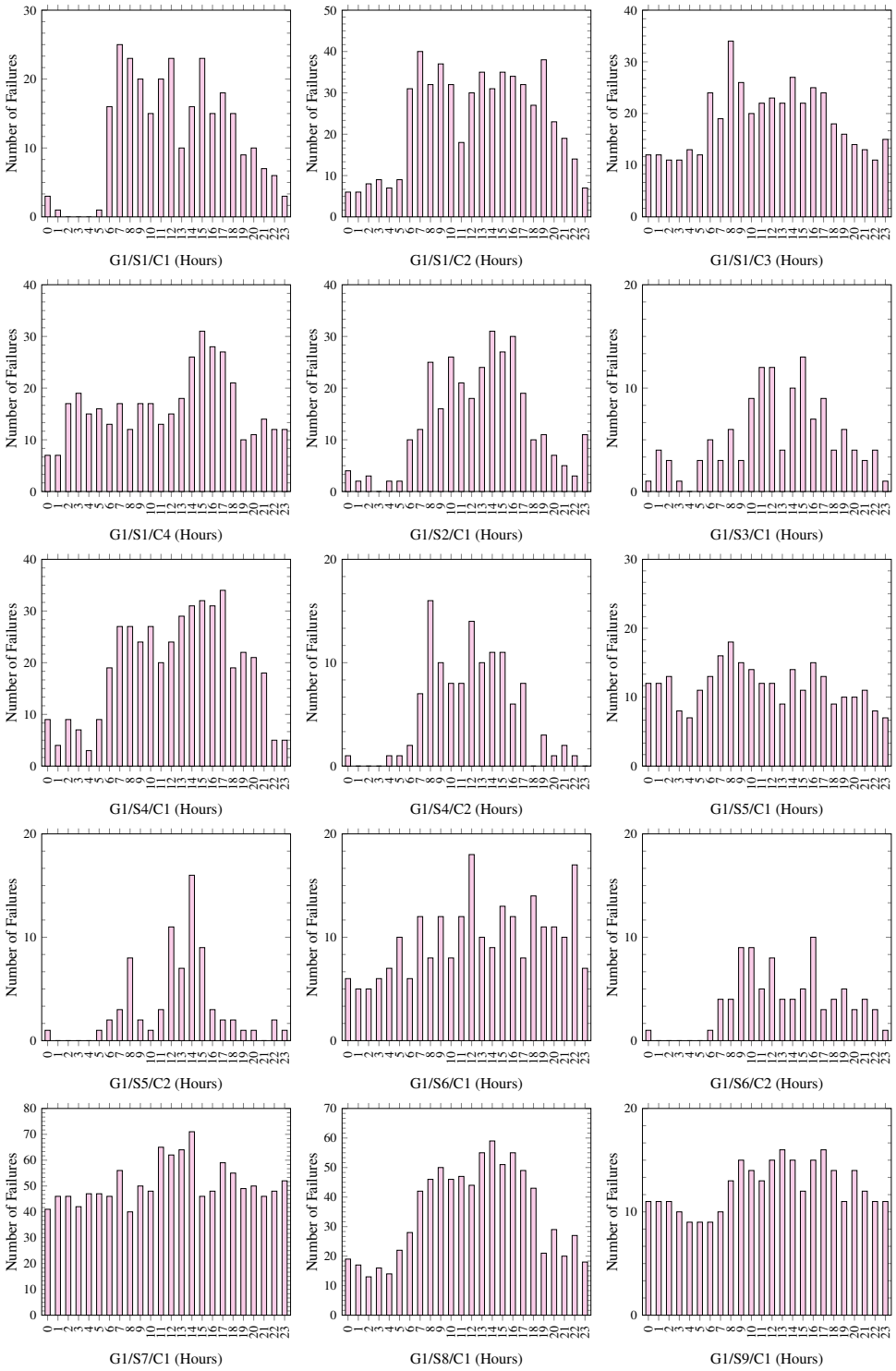


Figure 4.1: Failure Count vs. Daily Hours for Grid5000 Computing Systems

traces gathered from 26 different computing sites. The failure dataset consists of the data gathered from 9 geographically distributed sites consisting of 15 different clusters. Figure 4.1 shows the failure count per hour of a day corresponding to each node with maximum number of failure events per cluster present in the traces. After performing the analysis, a behaviour similar to chapter 3 is observed for the occurrence of failures w.r.t the time of a day. Such that during the working hours of a day, i.e., 7 am to 7 pm, the number of failure increases. This behaviour establishes a correlation between the occurrence of failures and utilization/activity of the system, such that, during the working hours, the utilization/activity of the systems is higher than the non-working hours, which increases the failure rate of systems.

On the basis the obtained correlation between the system utilization and occurrence of failures, it is concluded that the failure rate of a system/node/host depends on its utilization level which is a function of time, i.e., varying according to time. It is assumed that a VM running on a physical resource/node only fails, when the node fails. The failure rate/hazard rate, λ_{jk} of a vm_j running on a node, n_k with utilization u_k is calculated as, $\lambda_{max_k} \times u_j^{\beta}$, where, λ_{max_k} is the hazard rate of a node n_k at maximum utilization. β (> 0) is the sensitivity factor which shows the sensitivity of the failure rate towards the utilization. Maximum hazard rate, λ_{max_k} of a node n_k is the inverse of the Mean Time Between Failures ($MTBF$) of the node, $\lambda_{max_k} = \frac{1}{MTBF_k}$, which is calculated by the RMS using the history of failure events. The $MTBF$ corresponding to each host is calculated by using the real failure traces acquired from Failure Trace Archive (FTA) [88]. The failure rate is assumed to be following the Poisson distribution [138] [170] and remains constant for each utilization level. By using hazard rate λ_{jk}, the probability with which a VM, vm_j will be able to finish the execution of all the running tasks is equal to the probability with which the longest task, l_{max_j} running on vm_j will finish the execution before the occurrence of a failure. This probability is called reliability of a VM. As stated earlier, all the VMs running on a node fails when node fails such that node and VMs share a serial relationship with each other. So the reliability of a node n_k while running m VMs is calculated as the product of the reliabilities of all the VMs as follows

$$R_k = \prod_{j=1}^{m} (\exp^{-\lambda_{jk} \times l_j^{max}}) \tag{4.2}$$

However, all the provisioned resources/nodes fail independently from each other. So the reliability of the whole system is calculated as the average of the reliabilities possessed by each node.

4.3.2 Power Model

Every node in the resource pool has the power profile information such as the power consumption at minimum utilization, p_{min} and power consumption at maximum utilization p_{max}. To calculate the power consumption by a VM vm_j with utilization u_j running on a node n_k, the following model [14] is used

$$P_k(u_j) = (frac_k \times P_{max_k}) + \Big((1 - frac_k) \times P_{max_k} \times u_j\Big) \tag{4.3}$$

where, $frac_k$ is the ratio of P_{max_k} and P_{min_k}. Utilization, u_j of a VM, vm_j is calculated as the sum of normalized lengths of tasks $\in \Gamma_j$ executing on VM. The power model used to calculate the energy consumption of the system in this work considers the power consumed by CPU only because it has been argued that CPU is the biggest consumer of the power among other devices such as memory units and storage systems [140].

4.4 Failure Prediction

Fault tolerance methods in cloud computing systems are divided into two classes: reactive and proactive methods [137]. In reactive methods, the whole effort is to recover the failed tasks as soon as possible with minimum overheads whereas in proactive methods, the emphasis is to avoid the occurrence of failures. In Chapter 3, a reactive fault tolerance method, checkpointing is used to recover the failed tasks. In this work besides checkpointing, a proactive fault tolerance mechanism, VM migration is adopted to run the tasks without failing by migrating the running VMs from an expecting to be failed physical resource to other healthy and more reliable resources. To identify the physical resources expecting to be failed and to schedule the VM migrations, failure prediction is used. In order to predict the failures, an average based time series analysis method called Exponential Smoothing [63] is used. The reason for choosing the average based prediction

method is the inconsistency and stationarity of the available data, which is collected from Failure Trace Archive [88]. On the basis of the patterns found in the traces, parametric models such as ARIMA models [114] were tried to fit to predict the occurrence of failures but the Mean Square Error (MSE) was found to be higher than MSE of average based method. In order to predict the occurrence of failures, the prediction of Time between Failures (TBF) is targeted. To predict n failures, predictions of $n-1$ TBFs is done.

Among all the average based prediction methods such as simple average, weighted average and k-period moving average [114], the exponential smoothing is chosen because it considers all the time series values to make a prediction with associated weights. Though this is true for weighted average as well, exponential smoothing has a formal method to calculate the weights corresponding to each contributing value of the time series. Suppose there is a set of n TBFs corresponding to a host n_k, $TBF_k = \{tbf_t \mid 1 \leq t \leq n\}$. So the forecast corresponding to a tbf_{t+1} by using exponential smoothing is calculated as follows

$$(tbf_k)''_{t+1} = \begin{cases} \alpha \times (tbf_k)'_t + ((1-\alpha) \times (tbf_k)''_t), & \text{if } n > 1 \\ (tbf_k)'_t & \text{otherwise} \end{cases} \tag{4.4}$$

where, $(tbf_k)'_t$ is the actual value of TBF between two consecutive failures at time t and $(tbf_k)''_t$ is the forecasted value obtained at time $t-1$ for time t for a node k. α is the smoothing constant i.e. $0 < \alpha < 1$. Value of $(tbf_k)''_1$ is taken as the simple average of the time series.

4.4.1 VM Migration based on Failure Prediction

As mentioned in section 4.3, VM migration is adopted as a fault tolerance mechanism which gets triggered on the basis of the prediction results. VM migration can further be divided into two types: stop-and-copy VM migration and live VM migration [31]. Due to less down time and less migration overheads, live VM migration is adopted in this work.

VM Migration Overheads

Although VM migration has many advantages over other reactive and proactive fault tolerance mechanisms, small overheads in terms of execution time get imposed on all the tasks corresponding to a VM because of the interruptions while performing migrations. The overheads can vary according to the configuration of a VM such that memory usage and type of application that VM is executing. Among several live VM migration approaches [103], the pre-copy migration approach is adopted because of its less migration and downtime overheads. In pre-copy migration, the memory pages of a running VM, vm_j gets copied to the destination host, iteratively. The approach works with an assumption that at some point the memory pages required to get copied will be small enough so that the vm_j can be stopped and migrated to the destination host. This ensures the minimum memory page errors and less downtime. The proposed migration overhead model is based on the model provided by Sherif et al. [4]. The total migration overhead, TMO_{ij} for a task t_i running on vm_j is the sum of the migration overheads, MO_{ij} incurred during n pre-copy iterations and downtime overheads, DTO_{ij}.

$$TMO_{ij} = MO_{ij} + DTO_{ij} \tag{4.5}$$

The migration overhead, MO_{ij} is calculated as the sum of time taken by n pre-copy iterations and pre-migration overheads (Equation 4.6). The number of pre-copy iterations depends upon the page modification rate of an application running on VM. In this work, it is assumed that the considered BoTs application is running with high memory page modification rate. This makes the pre-copy iterations adding maximum overheads which is equal to n times of the VM size, vm_j^{size} less 1 page plus pre-migration overheads. Pre-migration overheads, ($PMO^{\diamond}$) are the overheads incurred before the migration starts such that resource reservation and migration initiation.

$$MO_{ij} = PMO^{\diamond} + \left(\frac{n \times vm_j^{size} - (P_{size})}{L_{speed}} \right) \tag{4.6}$$

vm_j^{size} corresponding to a VM is calculated as the product of total memory allocated to the VM, vm_j^{mem} and utilization of the VM, u_j. Downtime overhead, DTO_{ij} for a task t_i of VM, vm_j is

calculated as the sum of time taken to migrate the entire copy of the image of a VM and post-migration overheads, $PMO^{\sim}$ (Equation 4.7). Post-migration overheads are the overheads incurred during the re-activation of a migrated VM.

$$DTO_{ij} = \left(\frac{vm_j^{size}}{L_{speed}} \right) + PMO^{\sim} \tag{4.7}$$

It is assumed that $PMO^{\diamond}$ and $PMO^{\sim}$ are independent from the VM size and link speed, L_{speed}. So the overhead values remained constant during each VM migration. The proposed VM migration model is applicable to VM consolidation as well.

Task Finishing Time with VM Migration Overheads

When a VM, vm_j running on a physical host, n_k executing a set of tasks Γ_j gets migrated, the length of each task in the set Γ_j changes because of the migration overheads. Apart from the migration overheads, occurrence of failures also impact the execution time or finishing time a task. T^* represents the time required to re-execute the part of a task, which is equal to the part of task length executed before the occurrence of a failure. Besides the migration overheads and re-execution part of a failed task, time to return (TTR) from a failed state to running state also adds to the finishing time of a task. So the finishing time of a task t_i of length l_i executing on a vm_j with n migrations and m failures is calculated as follows

$$F_{ij} = \begin{cases} l_i + \sum_{p=0}^{n} TMO_{(ij)_p} + \sum_{q=0}^{m} TTR_{(ij)_q} + \sum_{q=0}^{m} T^*_{(ij)_q}, & \text{if } n, m > 0 \\ l_i & \text{otherwise} \end{cases} \tag{4.8}$$

In the absence of VM migration, the only factors contributing to the finishing time of a task are the re-execution part of a task (T^*) and time to return (TTR) from the failed state to the working state.

4.4.2 Checkpointing based on Failure Prediction

As a reactive fault tolerance mechanism, checkpointing is adopted in this work. Fault tolerance obtained by adopting checkpointing comes with huge cost in terms of task execution overheads.

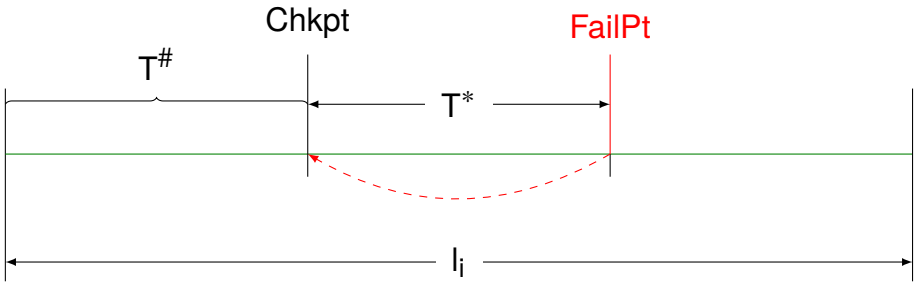


Figure 4.2: Recovery from Failure with Checkpointing

For instance, checkpointing adds the overhead of 151 hours for a job of length 1000 hours in a petaflop systems [119]. However, in the case of failure occurrence, re-execution of the failed task from the last checkpointing saves good amount of time. In general, events to create and save checkpoints get triggered with regular intervals (Section 3.4.1). However, in this work checkpoints are created according to the failure prediction results in order to optimize the checkpointing overheads and to bring the evaluation in accordance with VM migration. In order to further optimize the checkpointing overheads, risk based checkpointing is used such that if the expected amount of lost work before the checkpoint is smaller than the cost of checkpoint then skip the checkpoint [116].

VM Checkpointing Overheads

Similar to VM migration, the checkpointing overheads, T''_{ij} for a task, t_i executing on VM, vm_j varies according to its utilization, u_j. The overheads are calculated as the product of maximum overhead imposed by a checkpoint, CO^{max} and VM utilization.

$$T''_j = CO^{max} \times u_j \tag{4.9}$$

Task Finishing Time with VM Checkpointing

When a checkpointing event gets triggered for a VM, vm_j executing a set of tasks Γ_j, the current state such that the executed length ($T^{\#}$) of all the running tasks gets saved as a backup (Figure

4.2). The storage where all the backups are stored is assumed to be failure free. While saving the checkpoints, the length of all the tasks in the set Γ_j changes because of the checkpointing overheads (Equation 4.12). Besides the checkpointing overheads, other major factors that contribute to the finishing time of a task is the re-execution of the lost part of the task. T^* represents the time required to re-execute the lost part of a task from the last checkpoint because of a failure. Besides the checkpointing overheads and re-execution part of the failed task, time to return (TTR) from the failed state to running state also adds to the finishing time of a task. So the finishing time of a task t_i of length l_i executing on a vm_j with n checkpoints and m failures is calculated as follows

$$F_{ij} = \begin{cases} l_i + \sum_{p=0}^{n} T''_{(ij)_p} + \sum_{q=0}^{m} TTR_{(ij)_q} + \sum_{q=0}^{m} T^*_{(ij)_q}, & \text{if } n, m > 0 \\ l_i & \text{otherwise} \end{cases} \tag{4.10}$$

4.5 Energy Consumption Model

As the execution time of a running task changes because of failures, VM migration and checkpointing overheads (Equation 4.8-4.10), energy consumption by the system while executing the task get affected. The energy consumption by a VM, vm_j with utilization u_j running on a failure prone node n_k while executing a task t_i of length l_i is calculated as the sum of the energy consumed to execute the actual length of the task and energy wasted to execute the overheads.

$$E_{vm_{ij}} = \left(P_k\left(u_j\right) \times l_i\right) + E_{waste_{ij}} \tag{4.11}$$

But all the overheads that increase the execution time of a task do not contribute to the energy wastage. Among the factors such as TMO, T'', TTR and T^* that are considered to formulate the task finishing time in Equation 4.8 and 4.10, TTR did not contribute to the energy wastage because during the down-time, a system is in non-working state. There are many other precise details that are taken into consideration in the following subsections to formulate the energy wastage in order to predict the actual energy consumption.

4.5.1 Energy Wastage with VM Migration Overheads

To calculate the energy wastage while using VM migration as a fault tolerance mechanism, the energy wastage is further splitted into two parts i.e. energy wastage due to VM migration overheads and energy consumption to re-execute the lost part of the task because of failures. According to Equation 4.5, total migration overheads are the sum of migration overheads, MO and downtime overheads, DTO. MO does not contribute to the energy wastage, as during the migration, VM is in transition state and not running on any physical machine. So energy wastage is calculated as

$$E_{waste_{ij}} = E_{DTO_{ij}} + E^*_{ij} \tag{4.12}$$

But downtime, DTO is different from the downtime faced by a VM due to the occurrence of a failure i.e. TTR. During this downtime, a VM gets migrated to some other provisioned node with enough idle resources or a new node gets provisioned to accommodate the VM. During this process, an idle CPU core with 10% or less utilization, u_{idle} gets provisioned but not activated. The resource gets activated once the VM migration is completed. So the duration between after provisioning and before activation of a CPU core with u_{idle} is considered as the contributor to E_{waste} and is calculated as follows

$$E_{DTO_{ij}} = \begin{cases} (P_k\,(u_{idle})) \times \sum_{p=0}^{n} DTO_{(ij)_p}, & \text{if } n > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.13}$$

4.5.2 Energy Wastage with Checkpointing Overheads

For checkpointing, energy wastage for a VM, vm_j running on a node, n_k while executing a task, t_i is splitted into the energy consumption while saving the checkpoints and energy consumption while re-executing the lost part of a task.

$$E_{waste_{ij}} = E_{T''_{ij}} + E^*_{ij} \tag{4.14}$$

The power consumption while creating and saving checkpoints P_{chkpt} is found to be higher by 9 to 11% than the idle power $P\,(u_{idle})$ and much lower than the power consumption during the

execution of a task [106] [47]. This is because during the creation of checkpoints the activity of CPU decreases (biggest energy consumer) and activity of I/O controllers i.e. Direct Memory Access (DMA) controller increases in order to perform read/write operations on backup drives. So the energy wastage for task t_i running on a VM vm_j while using checkpointing is calculated as follows

$$E_{T''_{ij}} = \begin{cases} \left(f_{chkpt} \times P_k\left(u_{idle}\right)\right) \times \sum_{p=0}^{n} T''_{(ij)_p}, & \text{if } n > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.15}$$

where f_{chkpt} is the fraction of idle power consumed during a checkpointing operation. For both VM migration and VM checkpointing, the energy consumption while re-executing the lost part of a task due to the occurrence of m failures is calculated as follows

$$E^*_{ij} = \begin{cases} P_k(u_j) \times \sum_{q=0}^{m} T^*_{(ij)_q}, & \text{if } m > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.16}$$

By using Equation 4.16, energy wastage without VM migration and VM checkpointing can also be calculated because the only overhead that contributes to the energy wastage while not using VM migration or VM checkpointing is the task re-execution part. So the total energy consumption by x provisioned nodes allocated to m VMs while finishing all the tasks of BoT application in the presence of the occurrence of failures, using VM migration and VM checkpointing as the fault-tolerance mechanisms is calculated as

$$E_{total} = \sum_{k=1}^{x} \sum_{j=1}^{m} E_{vm_{jk}} \tag{4.17}$$

4.6 Resource and VM Management

Given the set of tasks and failure prone resources, the problem is how to provision the resources and allocate the VMs executing the tasks to maximize the reliability and minimize the energy consumption while keeping the number of provisioned resources minimum and ensuring every completion before the deadline.

Table 4.1: Nomenclature used in algorithms

Notation	Description
t_i	i_{th} task
l_i	Length of i_{th} task
l_{max}	Length of a longest task in T
r_k	k_{th} node
$\Re$	List of Provisioned nodes, $r_k \in \Re$
vm_j	j_{th} virtual machine
u_j	Utilization corresponding to each VM
C_h^*	Number of Idle cores on a target host, H
C_k^*	Number of Idle cores on a destination host, R_k
$T_h^{\sim}$	Next Prediction Time for Target Host, H
$T_k^{\sim}$	Next Prediction Time for Destination Host, R_k
rel_{jk}	Reliability of j_{th} virtual machine on k_{th} node
pow_{jk}	Power consumption of j_{th} virtual machine on k_{th} node

Before provisioning the physical resources, first the problem is to identify the number of VMs required to instantiated to execute the tasks of a BoT, B. This problem is formulated as bin-packing problem, where a VM is considered as a bin and capacity of a bin is the utilization of VM. Before instantiating new VMs, VM provisioning and task allocation algorithm (Algorithm 1) checks the available capacity on the running VMs and tried to allocate maximum number of tasks. For the remaining tasks, minimum number of new VMs to be instantiated is calculated using Equation 3.17. After calculating the minimum number of VMs, physical machines from the pool of heterogeneous resources are provisioned (Algorithm 3). The physical resources in the pool have different hazard-rates and power profiles and are chosen on the basis of the objective of the cloud provider to minimize the energy consumption or maximize the reliability or achieve both objectives at the same time. As the focus of this work is to maximize the reliability and minimize the energy consumption together, Reliability-Energy Aware Best Fit Decreasing (REABFD) resource provisioning and VM allocation policy (function 3) is chosen to rank the physical resources by using the ratio of MTBF and power consumption corresponding to current utilization of each resource. Once the physical resources are provisioned and allocated to VMs, an optimized allocation of remaining tasks is done while taking the status of all the running VMs into consideration (Algorithm 1).

Algorithm 3: Resource Provisioning and VM Allocation

Input: $List\ of\ Resources, \mathbf{R}\ and\ List\ of\ VMs, \mathbf{V}$
Output: $Set\ of\ Provisioned\ Resources\ and\ Allocated\ VMs$

```
1  R_sorted ←ReliabilityAndEnergyAware(R);
2  for j ∈ V do
3      VM_cores_j ← vm_j.coresRequired();
4      for k ∈ ℜ do
5          if ((ℜ_k == failed)) then
6              continue;
7          else
8              if ((ℜ_k.predictedtoFail()! = true)&&(RC_k ≥ VM_cores_j)) then
9                  r_k ← vm_j.allocateHost();
10                 RC_k = RC_k − VM_cores_j;
11                 //Calculate VM reliability (Equation 4.2);
12                 R_vm_jk ← vm_j.calculateReliability();
13                 //Estimate VM power consumption (Equation 4.3);
14                 pow_k(u_j) ← vm_j.estimatePower();
15                 if (RC_k == 0) then
16                     R_sorted = R_sorted − ℜ_k;
17                 break;
18 //In case of unallocated VM, provision new host from pool, P if
   (vm_j.unallocated() == true) then
19     ℜ ← R_sorted;
20     Goto 4
```

While allocating VMs to physical machines, RMS first checks the availability of idle cores on the provisioned and active physical machines $\in \mathfrak{R}$ and allocate VMs to the idle cores, if available (step 4-17). If the resource requirement for a VM, vm_j is not fulfilled by physical machines $\in \mathfrak{R}$, then a new physical machine is provisioned from P (step 18). To tolerate or avoid the occurrence of failures, failure prediction is used in order to trigger the fault tolerance mechanisms (Equation 4.4). Once the fault tolerance mechanism either reactive or proactive is triggered for a node, the node is labelled as $predictedtofail$ and was not available for the allocation to new or migrating VMs (Algorithm 4) until the label is removed (step 8). The label is removed after the predicted

failure is occurred. In the case of checkpointing, VM remains on the same node through out its lifetime. On the basis of the failure prediction results, the present state of all the running tasks on VM gets saved so that in case of a failure, they will get recovered from the last checkpoint. The failing node is labelled as $predictedtofail$ and was not available for further allocation until the label is removed.

Algorithm 4: Failure Aware VM Migration

Input: $Predicted\ to\ be\ Failed\ Resource$, **R**
Input: $List\ of\ Resources$, **R** $and\ List\ of\ VMs$, **V**
Output: $New\ allocation\ for\ VMs$

```
1 //Identify a resource with the earliest predicted failure time;
2 V ← R_k.VmList();
3 for j ∈ V do
4     R_k.deallocate(vm_j);
5     vm_j.setInMigration();
6     //Algorithm 2 is called to select new host;
7     resourceProvisioningAndVMAllocation(vm_j);
8     //Add overheads using Equation 4.8;
9 R_k.predictedtoFail(true);
```

After finishing all the allocated tasks, a VM gets destroyed. Its corresponding resources sit idle and contribute to energy wastage unless these resources are allocated to a new VM. In order to reduce such idle resources, VM consolidation (Algorithm 5) is adopted, which get triggered when a VM is destroyed. The proposed method is a failure-aware VM consolidation policy in which the maintenance of reliability of the system is targeted while minimizing the energy wastage by considering the failure characteristics of current host (host on which VM is destroyed) and target host, which is chosen to perform consolidation. While performing the VM consolidation, new physical resources were not provisioned and VMs get consolidated between the provisioned physical resources. Before performing VM consolidation, RMS needs to decide whether VMs will get consolidated to the current host (host on which VMs are currently running) or out of the current host. This is decided while selecting a target host. The selection of a target host is made on the basis of two criterias. First criteria is the number of idle cores. If the number of running VMs

Algorithm 5: Failure Aware VM Consolidation

Input: $List\ of\ Resources, \mathbf{R}\ and\ Targest\ Host, \mathfrak{R}_h$
Output: $New\ host\ and\ set\ of\ VMs$

```
1  V_h ← ℜ_h.VmList();
2  C*_h = ℜ_h.numberofIdleCores();
3  T~_h = ℜ_h.nextFailurePredictionTime();
4  for k ∈ ℜ do
5      T~_k = ℜ_k.nextFailurePredictionTime();
6      C*_k = ℜ_k.numberofIdleCores();
7      V_k ← ℜ_k.VmList();
8      if ((C*_k ≥ V_h.size()) && (T~_h < T~_k)) then
9          if ℜ_k.predictedtoFail() == true then
10             continue;
11         flag == true;
12         for j ∈ V_h do
13             ℜ_h.deallocate(vm_j);
14             vm_j.setInMigration();
15             vmAllocation(vm_j, ℜ_k);
16             //Add overheads using Equation 4.8
17     if ((C*_h ≥ V_k.size()) && (T~_k < T~_h)) then
18         if (ℜ_h.predictedtoFail() == true) then
19             continue;
20         for j ∈ V_k do
21             ℜ_k.deallocate(vm_j);
22             vm_j.setInMigration();
23             vmAllocation(vm_j, ℜ_h);
24             //Add overheads using Equation 4.8
25     if (flag == true) then
26         //Turning off the target host;
27         setState(ℜ_h, off);
28         ℜ = ℜ − ℜ_h;
29         break;
30     else
31         //Turning off the destination host;
32         setState(ℜ_k, off);
33         ℜ = ℜ − ℜ_k;
34         break;
```

Table 4.2: Simulation Configuration Parameters

Input Parameters	Values
Stringency Factor (f)	1.3
Sensitivity Factor (β)	1
Number of Pre-Copy Iterations (n)	4
Page Size (P_{size})	4KB
Pre-Migration Overheads ($PMO^{\diamond}$)	15.6403 secs
Post-Migration Overheads ($PMO^{\sim}$)	.9070 secs
Link Speed (L_{speed})	1Gbps
Smoothing Constant (α)	.9
Maximum Checkpointing Overhead (CO^{max})	20 secs
Idle Power Checkpointing Fraction (f_{chkpt})	1.15

on the current host is less than or equal to the idle cores on the target host, then the next failure prediction time of both hosts is considered as the second criteria. If the failure prediction time of the current host is smaller than the target host then all the running VMs from the current host get consolidated to the target host (step 8-16) and current host is turned off. However, if the number of running VMs on the target host is less than or equal to the idle cores available on the current host and failure prediction time of the target host is smaller than the current host, then all the running VMs from the target host get consolidated to the current host (step 17-24) and target host is turned off. In case, failure prediction values were not available, hazard-rates of current and target hosts are used.

4.7 Performance Evaluation

After doing a Matlab based verification of all the proposed mathematical models and algorithms, a simulation based validation of the same is done using a real cloud computing architecture purposed in Figure 3.1 and using configuration parameters given in Table 4.2. In order to simulate the architecture, a well known cloud computing simulator $'CloudSim'$ [27] is extended by adding failure injector, fault tolerance and VM consolidation.

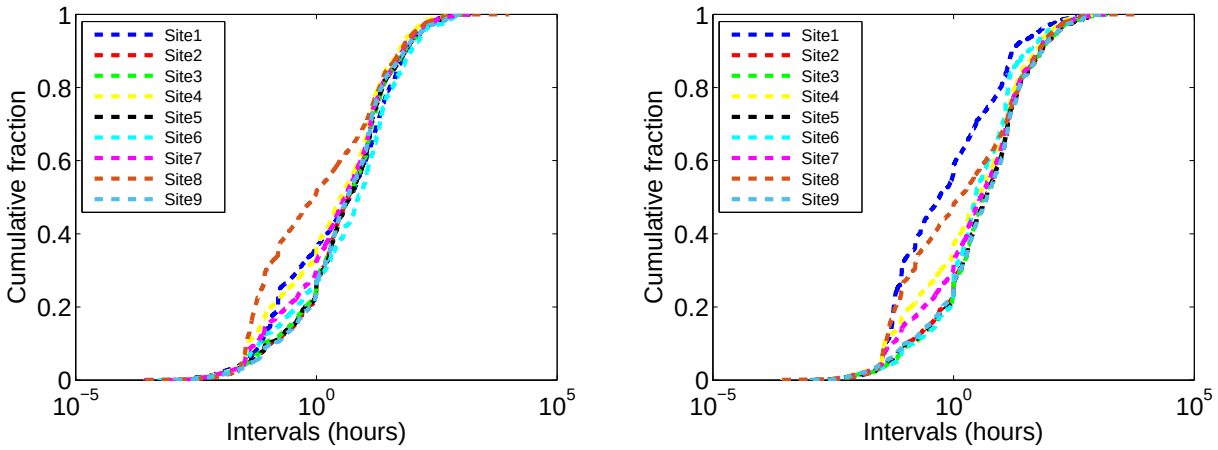


Figure 4.3: CDF of (a) Time between Failures (TBF) (b) Time to Return (TTR)

4.7.1 Simulation Setup

Grid5000 failure dataset which is collected for 1.5 years between 2005-2006 is downloaded from Failure Trace Archive (FTA)[88] and used in this work. Traces provided information about the failures and hardware configuration of approximately 1300 nodes. All the nodes are installed across 9 geographically distributed sites consisting of 15 different clusters. The mean time between failures (MTBF) and mean time to return (MTTR) for each node corresponding to each cluster are calculated by using the failure information provided in the traces. The cumulative distribution functions (CDFs) of time between failures (TBF) i.e. availability events and time to return (TTR) i.e. unavailability events for all the 9 different sites are given in Figure 4.3. From CDFs, same behaviour can been seen between the occurrence of availability and unavailability events. Parameter fitting is done for various distributions and both TBF and TTR events found to be following Weibull and log-normal distributions.

In order to choose the value of smoothing constant for the failure prediction (Equation 4.4), a statistical analysis of failure accuracy is conducted using different values of smoothing constant (Figure 4.4). The failure prediction accuracy is calculated as the percentage of failures predicted before the occurrence of failures. The analysis is conducted using nodes from each cluster of 9

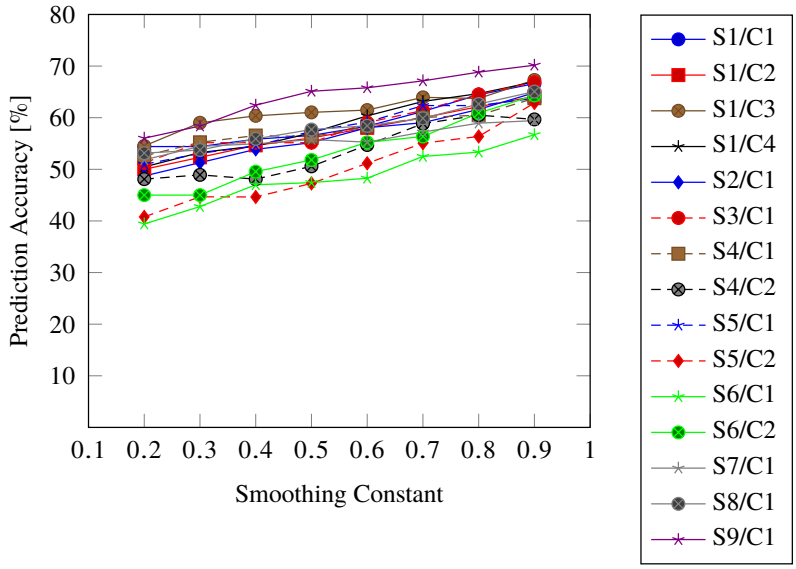


Figure 4.4: Prediction Accuracy vs Smoothing Constant. The analysis is carried out by changing the value of α in equation 4.4 from .2 to .9

different sites with the maximum number of failure events. From the analysis, it is observed that as the value of smoothing constant increases from 0.2 to 0.9, the failure prediction accuracy also increases. Consequently, the less number of past values contribute to the short term prediction, the better the prediction results are achieved. Same behaviour is observed by using moving average prediction method such that by using smaller window size better failure prediction accuracy is achieved. This is because of the interpolation prediction being performed such that for each failure event value in failure traces, a corresponding failure prediction value is generated. If the generation of failure prediction values beyond the number of values provided in failure traces (extrapolation) was required then the contribution of past prediction values (smaller smoothing constant value or larger window size) would have been more desirable. By using exponential smoothing with smoothing constant 0.9, the achieved prediction accuracy is between 57% to 71%.

While going in accordance to Xen hypervisor configuration of the system architecture (Section 3.3), the page size, P_{size} for pre-migration, $PMO^{\diamond}$ overheads is set to 4 KB as it is the minimum page size for Xen hypervisor. Link speed, L_{speed} is set to 1 Gbps in order to perform VM migrations without any transmission delay. To keep the deadlines corresponding to tasks moderate [78] and to

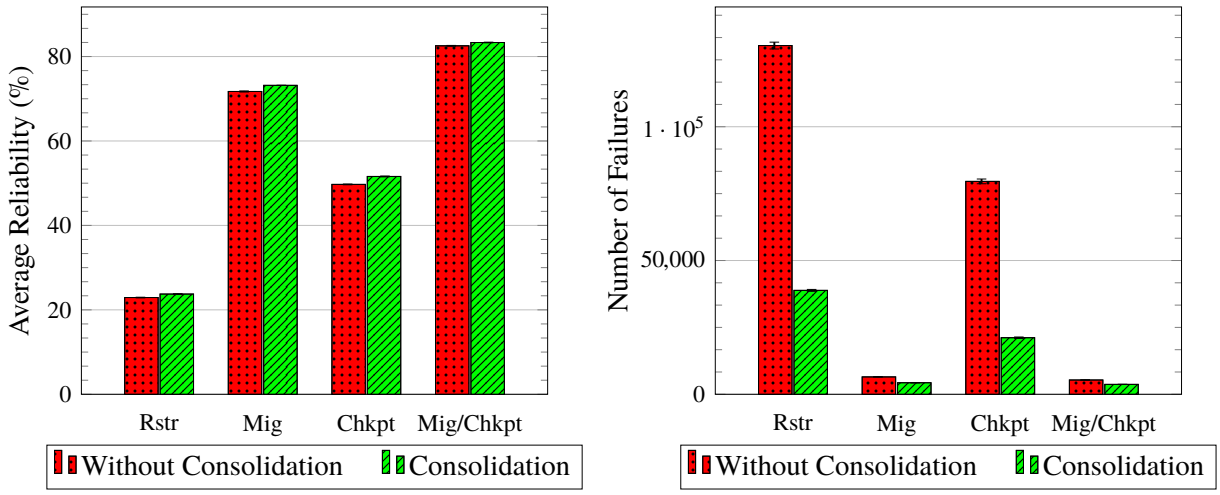
keep a strict linear relationship [174] between the utilization and reliability, values for stringency factor, f and sensitivity factor, β are set to 1.3 and 1, respectively. Idle power checkpointing fraction, f_{chkpt} is set to 1.15 [47] and other parametric values for VM migration overheads are obtained from [4]. To calculate the power consumption, the values of minimum and maximum power consumptions corresponding to a node are taken from spec2008 benckmark. To generate the BoTs workload, model proposed by Iosup et al. [75] is used with parameters given in Table 3.3.

4.7.2 Results and Discussions

Performance evaluation of the proposed resource provisioning and VM consolidation policy using different fault tolerance mechanisms is done in terms of reliability, finishing time and energy-efficiency. All the simulations are performed using 200 BoTs consisting of total number of tasks between 20000 to 25000. All the reported results are the average of 50 simulations with 95% confidence interval. For brevity, in the discussion of results below, 'Rstr' represents the base scenario of 'restart' in which no fault tolerance mechanism is used, and Chkpt, Mig and Mig/Chkpt represents scenarios with checkpointing, VM migration and combination of VM migration and checkpointing, respectively. ***Note:*** VM migration is used for two purposes such that for fault tolerance and for energy regulation (VM consolidation). Though the method is same, purpose and triggering criteria are different. In the case of fault tolerance (Algorithm 4), VM migration get triggered on the basis of failure prediction results. However, VM consolidation (Algorithm 5) get triggered when a VM is finished the execution of its corresponding tasks and get destroyed and corresponding resources become idle.

Reliability Evaluation

Figure 4.5a presents the average reliability of the system using various fault tolerance mechanisms with and without consolidation. For all the scenarios, it is observed that despite of the extra migration overheads imposed while performing VM consolidation, the system possessed higher reliability than the scenarios without consolidation. Which is achieved despite of the fact that by

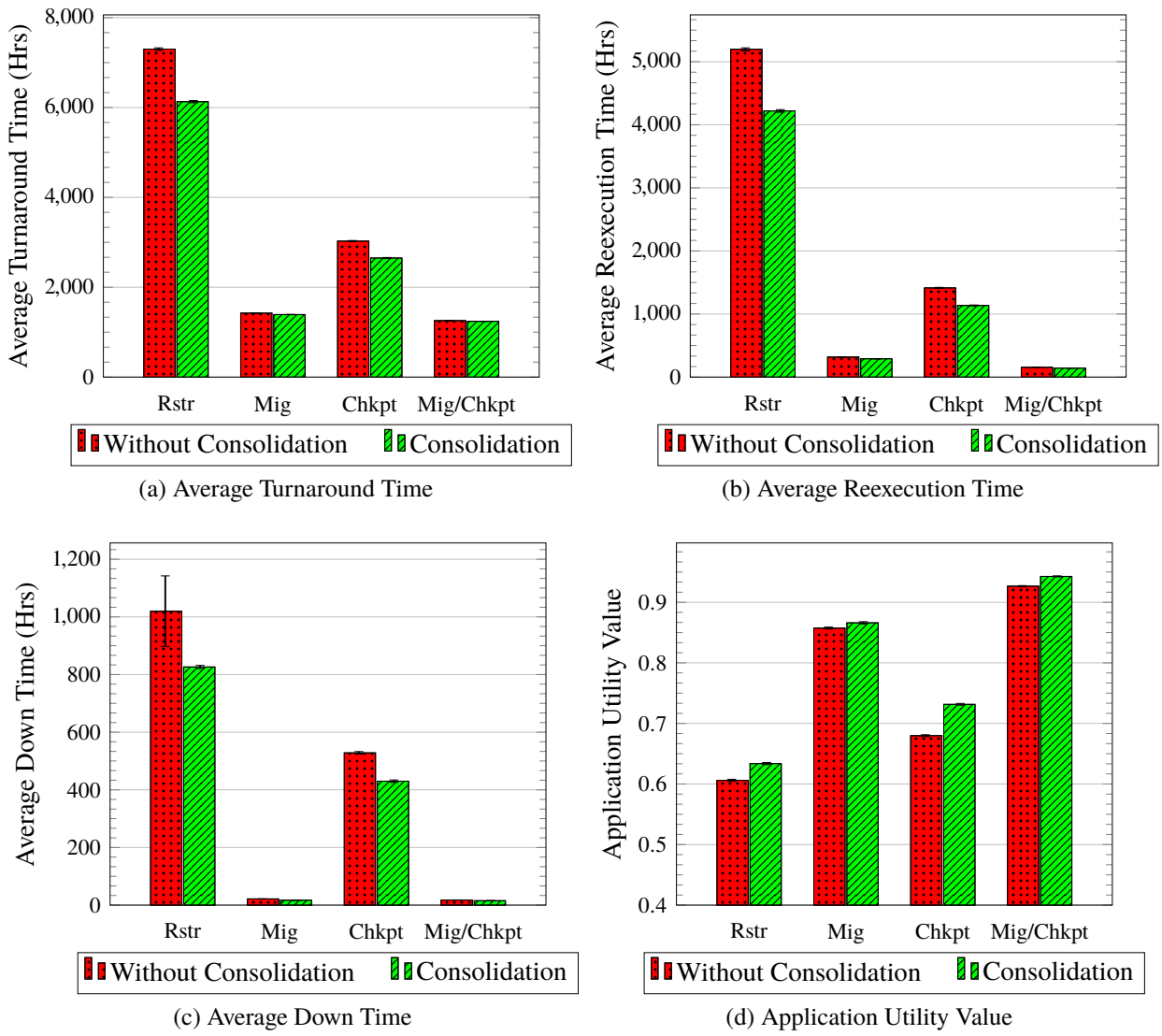


(a) Average Reliability

(b) Number of Failures

Figure 4.5: Results for Reliability Evaluation (Rstr: Restart, Mig: Migration, Chkpt: Checkpointing)

imposing extra overheads, task length increases which in return reduces the reliability of a system (Equation 4.2). However, the obtained results are because of a huge reduction of approximately 73% in the occurrence of failures (Figure 4.5b) for the scenarios using failure-aware VM consolidation in comparison to without VM consolidation scenarios. With the reduction of number of failures, the re-execution part of failed tasks is reduced significantly (Figure 4.6b), which complemented the VM consolidation overheads and increased the reliability of the system. Among the used fault tolerance mechanisms, it is found that VM migration (Mig) used as fault tolerance mechanism outperformed checkpointing (Chkpt) by reducing the failure count by up to 80%. This failure count is further reduced with adoption of 'Mig' in conjunction of 'Chkpt' (Mig/Chkpt) by 14% and 18% for both with and without consolidation environments, respectively. Such reduction of number of failures lead to significant changes in the reliability where the combination of VM migration with checkpointing used in failure-aware VM consolidation environment ensured maximum reliability among all the scenarios. ***Note:*** Although by using 'Mig' and 'Mig/Chkpt', attempt is made to avoid the occurrence of failures by migrating running VMs from suspicious nodes predicted to be failed to other healthy nodes, failures still happened because of the failure prediction errors.



(a) Average Turnaround Time

(b) Average Reexecution Time

(c) Average Down Time

(d) Application Utility Value

Figure 4.6: Results for Execution Time Evaluation

Execution Time Evaluation

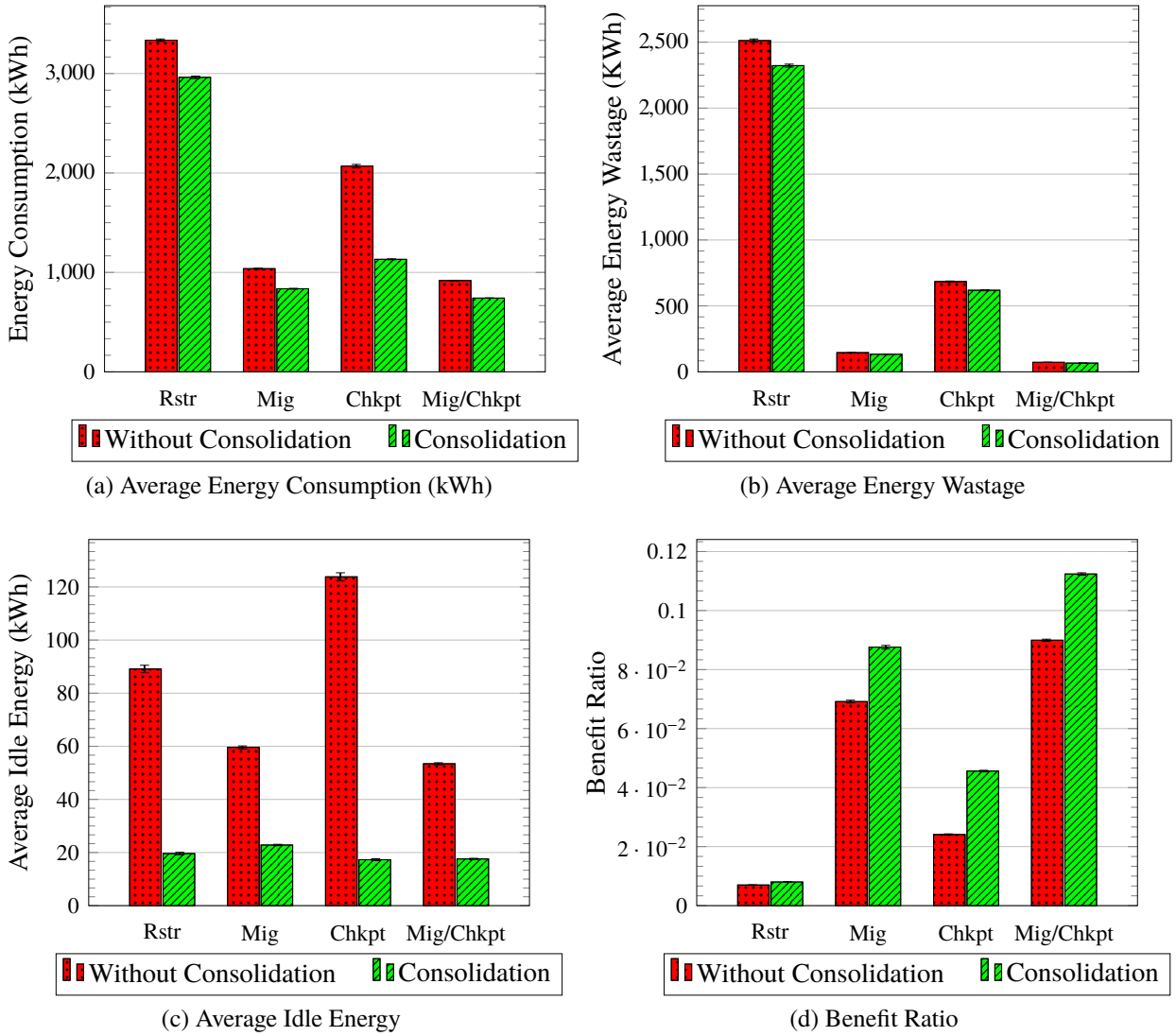
Figure 4.6a shows the average turnaround time that is achieved while executing the tasks of BoT application. From the figure it is observed that the 'Rstr' scenario experienced maximum task completion time for both with and without consolidation because of higher downtime (Figure 4.6c) and re-execution time (Figure 4.6b) due to large number of failures (Figure 4.5b). Among the scenarios using fault tolerance, 'Mig/Chkpt' scenario has minimum completion time such that lower

by 12% and 58% from 'Mig' and 'Chkpt', respectively because of less re-execution and downtime overheads. This completion time further improved by 12% with the adoption of failure-aware VM consolidation policy despite of the extra migration overheads (Equation 4.5-4.7) imposed while performing consolidation. This is because of the improvement obtained in terms of number of failures which eventually reduced the re-execution time and downtime and resulted in the reduction of the turn around time corresponding to running tasks (Equation 4.8).

The objective behind reducing the turnaround time is to finish the tasks before the corresponding deadlines (Equation 3.1) in order to achieve maximum utility value (Equation 4.1) and to ensure high quality of services (QoS). Figure 4.6d shows the achieved application utility value while using different fault tolerance mechanisms with and without VM consolidation. From the figure, it can be seen that maximum application utility value is achieved by using the combination of VM migration and checkpointing (Mig/Chkpt). This is because if a failure happened for 'Mig/Chkpt' scenario, the re-execution of a failed task started from the last saved checkpoint rather than starting from the beginning, which is happening while using only VM migration (Mig) as fault tolerance mechanism. Starting from the last checkpoint in 'Mig/Chkpt' scenario resulted in the lowest downtime (Figure 4.6c) and re-execution time (Figure 4.6b). An improvement of 7% over 'Mig' and 27% over 'Chkpt' is achieved without using consolidation. By introducing failure-aware VM consolidation the utility value is further improved by 1% for 'Mig', 2% for 'Mig/Chkpt' and 7% for 'Chkpt'. Such improvement of utility value shows that despite of extra migration overheads imposed while performing VM consolidation, more deadlines are achieved than without consolidation with a good improvement over the energy wastage (section 4.7.2) and reliability of the system.

Energy Efficiency Evaluation

Figure 4.7a presents the average energy consumption of the system. Obviously, system has consumed maximum energy under 'Rstr' because of the large re-execution overheads (Figure 4.6b) incurred due to the occurrence of large number of failures (Figure 4.5b). However, among the scenarios with fault tolerance mechanisms, 'Chkpt' consumed maximum energy such that while



(a) Average Energy Consumption (kWh)

(b) Average Energy Wastage

(c) Average Idle Energy

(d) Benefit Ratio

Figure 4.7: Results for Energy Efficiency Evaluation

using consolidation it is more by 26% from 'Mig' and 34% from 'Mig/Chkpt' and for without consolidation the increment is 50% from 'Mig' and 56% from 'Mig/Chkpt'. This is because of the reactive behaviour of checkpointing mechanism where the occurrence of failures is certain and recovery happens after the occurrence of a failure and to tolerate the impact of a failure, execution restarts from the last checkpoint. However, in VM migration based scenarios, the attention is also paid on the failure avoidance besides the failure tolerance. Which resulted in lesser re-execution time and system downtime because of lower failure count. Among the 'Mig' and 'Mig/Chkpt' scenarios, less energy consumption is experienced for 'Mig/Chkpt' scenario for both with and without consolidation. In all the cases, scenarios using consolidation found to be using less energy consumption than scenarios without consolidation despite of extra VM migration overheads (Equation 4.5-4.7). This is because by using the proposed failure-aware VM consolidation policy (Algorithm 5), less failure occurrence is experienced (Figure 4.5b) which in return reduced the re-execution time (Figure 4.6b) of tasks.

The main objective of using VM consolidation is to reduce the number of underutilized resources by turning them off or by putting them on sleep/hibernation mode after migrating the corresponding VMs to other underutilized resources. By doing this, the resource efficiency and utilization increases and reduces the idle energy consumption by either idle resources or underutilized resources. Figure 4.7c shows a significant reduction in idle energy consumption of the system while performing the proposed failure-aware VM consolidation policy. Such that, the idle energy consumption is brought down by 86% for 'Chkpt', 67% for 'Mig/Chkpt', 62% for 'Mig' and 78% for 'Rstr'. Among all the fault tolerance scenarios, 'Mig/Chkpt' consumed minimum idle energy which is lesser by 23% from its nearest rival scenario i.e. scenario using 'Mig' because of lesser failure count (Figure 4.5b) and turnaround time (Figure 4.6a). Lower the turnaround time, early the tasks will finish the execution, which will proceed to turning the idle or underutilized resources off earlier.

As the objective of this work is to improve the reliability and energy efficiency jointly, a metric called Benefit Ratio (the ratio of achieved reliability and energy consumption) is introduced to reflect the overall improvement of proposed mechanisms (Figure 4.7d). While being consistent with previous results, the combination of 'Mig' and 'Chkpt' gives the best benefit ratio value which

is higher by 23% than the scenario using 'Mig' as the fault tolerance method. This value is further improved by 15% with the adoption of proposed failure-aware VM consolidation policy.

4.8 Summary

In this chapter, a failure-aware energy-efficient VM consolidation policy is presented. It takes the reliability factor into consideration before consolidating the running VMs in order to save energy in a failure prone cloud computing environment. To provide fault tolerance, both reactive (checkpointing) and proactive (VM migration) mechanisms are used. To trigger the fault tolerance mechanisms, time series analysis based failure prediction is introduced. Verified by extensive simulation study, the following conclusions are drawn

1. While performing the VM consolidation in a failure prone cloud computing system, a significant improvement in terms of energy efficiency and reliability can be achieved by considering the failure characteristics of physical resources.

2. To achieve higher fault tolerance in cloud computing systems, it is better to use the combination of reactive and proactive fault tolerance mechanisms rather than using them individually.

The failure-aware VM consolidation policy proposed in this chapter considers the independent occurrence of failures. However, the occurrence of a failure can trigger failures in other components in cloud computing infrastructure in a correlated manner. In order to deal with such scenarios, correlated failure-aware resource provisioning and VM consolidation mechanisms for cloud computing systems are proposed in the next chapter.

Chapter 5

Reliable and Energy Efficient Cloud Computing Systems under Correlated Failures

This chapter proposes many mechanisms for jointly improving reliability and energy efficiency under correlated failures in cloud computing systems. To well understand failure correlation, statistical cluster analysis techniques are applied to real failure traces. To predict the occurrence of failures, an average-based time series analysis, i.e., moving average method is used. Then, mathematical models are built to represent reliability and energy consumption of cloud computing systems in the presence of correlated failures. These mathematical models are used to design fault-tolerant and energy-aware resource provisioning mechanisms/policies. In order to further reduce the energy consumption, a correlated failure-aware VM consolidation policy is also proposed in this chapter. A simulation based study of the proposed resource management policies and fault tolerance mechanisms is conducted by using real failure traces. The results demonstrate that by exploiting failure correlation with the proposed resource management policies, the occurrence of failures is reduced by 34% and increased the energy efficiency of the system by 20% while improving the reliability by 10%, approximately.

5.1 Introduction

Dependence of computing resources on each other in cloud computing systems makes them prone to fail in a correlated manner which impacts service reliability and energy efficiency of the system. The reason for the occurrence of such failures is the failure of shared resources. The occurrence of correlated failures in cloud computing systems lead to a service outage and violation of Service Level Agreement (SLA), which costs huge to cloud service providers and users. In October 2013, Knight Capital's cloud based automatic stock trading system went down for 45 min because of a software error which costed $440 million to the company [21]. A report released by Ponemon institute in 2016 revealed that the average down-time cost of data centers due to outages is approximately $740,357 per year [73]. Such huge costs do not occur due to the failure of a single computing resource. It mainly happens because of the outage of multiple resources in a correlated manner. In June 2017, section of Sydney based AWS data center suffered with a power loss resulted in the termination of multiple number of instances [7], which represents the occurrence of a failure in correlated manner. Similar kind of incidents representing the occurrence of correlated failures took place for other services or applications mounted on cloud computing infrastructure such as Microsoft Azure [43] and Google Compute Engine [66].

Failures either independent or correlated in physical resources are inevitable and can happen due to multiple reasons such as hardware outage, software glitches etc. [137]. It has been argued that a system with 100,000 processors experience a failure every couple of minutes [50]. To tolerate the occurrence of resource failures, cloud computing providers have adopted many fault tolerance techniques such as resource redundancy [22] and replication [173] and load balancing [137]. But improving the reliability of cloud computing services using redundant resources increases the energy consumption severely, which is already an existing challenge. It has been reported that servers mounted in Microsoft's cloud based data centers consume approximately 2 terawatt-hours (TWh) of energy per year for which the company pays approximately $2.5 billion per year as electricity bills [9]. Most of the servers in such data centers are sitting idle and are deployed to accommodate peak load in order to avoid an outage [158].

The energy consumption can be saved by reducing the number of active but idle or underutilized resources, which on the contrary reduces the reliability of the system. This creates a critical trade-off between these two metrics which provided motivation to focus on regulating the energy consumption of cloud computing systems while tolerating the occurrence of correlated failures. In particular, the main contributions of this chapter are as follows

- A statistical methodology is provided to find the correlation between the occurrence of failures. The failure correlation information is further used to create homogeneous groups called clusters of physical resources sharing common failure characteristics. To create such clusters, statistical cluster analysis techniques are used. Parameter estimation techniques are also proposed for the cluster analysis.
- Mathematical models are proposed to calculate the reliability, task finishing time and energy consumption while taking the overheads imposed by the occurrence of failures and fault tolerance mechanisms into account.
- The cluster information and mathematical models are used to design various resource provisioning and VM allocation algorithms. The focus is mainly paid on the problem of reliability aware and energy efficient resource management in the presence of correlated failures. The proposed resource management policies are equipped with reactive (checkpointing) and proactive (VM migration) fault tolerance mechanisms to tolerate or mitigate the occurrence of correlated failures. In order to regulate energy consumption dynamically, a correlated failure-aware VM consolidation policy is also proposed.

5.2 Related Work

The occurrence of failures in distributed computing systems i.e., cloud computing systems can be categorized into two classes, independent failures and correlated failures [137]. Furthermore, the correlation between failures can be modeled as temporal correlation [166] and spatial correlation

[56]. Because of the shared resources in cloud computing infrastructure, some times the occurrence of a failure in one part gets propagated, which triggers the series of failure events in the whole system. Due to the existence of such scenarios, the assumption of independent failures is unrealistic. However, most of the works carried out to increase the fault tolerance in cloud computing systems ignores the correlated occurrence of failures [110]. This makes the proposed solutions some time non-applicable to the scenarios with correlated failures. It has been argued that ignoring the correlated failures can over estimate the availability of a system by at least two orders of magnitude [52]. To find the correlation between failures in distributed computing systems, different methods and models [172] [118] are proposed in the literature but their application demonstration in terms of allocation and scheduling is very limited. Furthermore, very limited works are done considering failure correlation in cloud computing.

The current but limited research has shown that by taking failure correlation into account, researchers have achieved better performance and fault tolerance in cloud computing systems. Mina Sedaghat et al. [135] targeted a problem to schedule the tasks in a reliability manner with minimum number of replicas in a cloud computing system with correlated failures. Authors have presented a statistical reliability model for reliability estimation and incorporated it in a correlated-failure aware task allocation policy. The considered correlated failures are assumed to be triggered by power and network outages. However, in this work correlated failures are independent from the reason of occurrence. To measure the likelihood of correlated failures, authors have used affinity score metrics [52] where as cluster analysis techniques using real failure data are used in this work to handle the correlated failures. Unlike the static fault tolerance solution such as replication used by authors, more dynamic and energy efficient fault tolerance solutions such as checkpointing, VM migration and VM consolidation using failure prediction results are used in this work.

With the same objectives while targeting network failures, Ao Zhou et al. [173] also adopted VM replication technique based on K-fault tolerance replication to enhance the service reliability in cloud computing systems. Similar to this work, authors have targeted to minimize the number of working resources while enhancing the reliability to reduce the operational expenses. Authors did not highlight the correlated failures as such but concerning about the placement of VMs on the

hosts sharing same network resources can be taken as a correlated failure aware VM allocation. Authors have proposed a static VM allocation policy by placing VM and corresponding K replicas on different hosts belongs to different network domains. Though the proposed methods provide high fault tolerance using replication, the cost in terms of resource usage and energy consumption is ignored. However, this work has used other fault tolerance mechanisms such as checkpointing and VM migration because of high usage cost of replication while ensuring high reliability and minimizing energy consumption in a correlated failure aware cloud computing system. This work has also explored heterogeneity and dynamic nature of cloud computing systems by employing different hardware types and VM consolidation and also used real time failure traces to inject failures in a simulated cloud computing environment. Similar kind of study using the outage of power supply and its impact on other components in cloud computing systems is done by Elisson Rocha et al. [129]. Similar to the previous works, authors used redundant resources to increase the availability of the system without considering ownership cost and energy efficiency of the system.

5.3 Failure Correlation and Prediction

In distributed computing systems, it has been argued that the occurrence of a failure can trigger simultaneous or successive events of failures within a short interval of time [118]. In this work, simultaneously occurred failures are identified as correlated failures. In order to tolerate such occurrence of failures, it is very important to explore any kind of correlation existing between the failures. By identifying the correlation, the heterogeneous physical nodes are grouped into clusters on the basis of their availability such that time between failures (TBF) characteristics using statistical cluster analysis techniques. Such formation of clusters is further exploited to tolerate or avoid the occurrence of failures. Consequently, the triggering of failure tolerance or avoidance measure for one host in a cluster will lead to the triggering of the same measure for the rest of nodes in the cluster. The failure tolerance or avoidance measures are triggered on the basis of failure prediction results obtained by using an average-based time series analysis method (subsection 5.3.3).

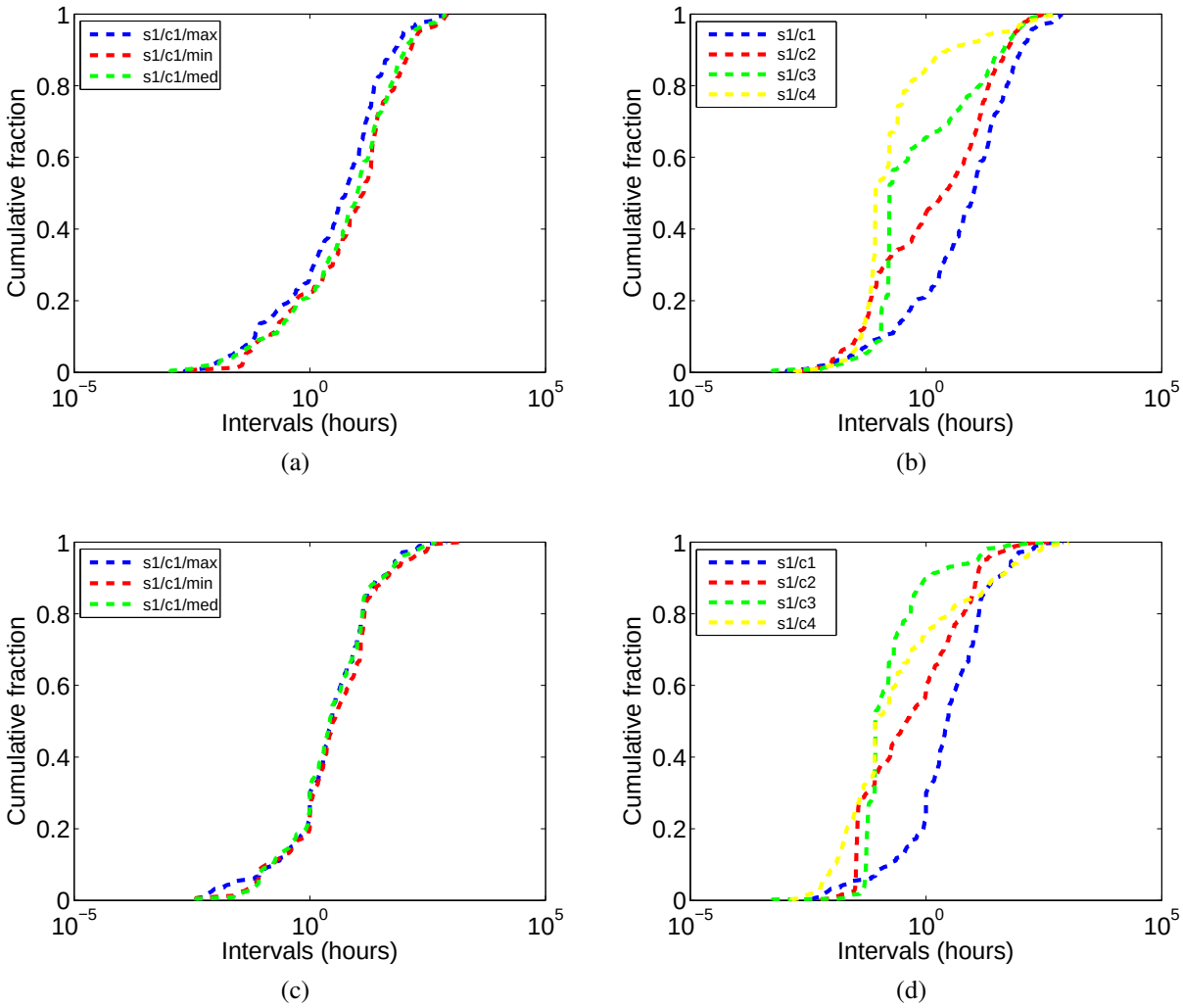


Figure 5.1: CDF of (a) Intra-Cluster Node Availability (b) Inter-Cluster Node Availability (c) Intra-Cluster Node Unavailability (d) Inter-Cluster Node Unavailability

5.3.1 Failure Correlation

The failure correlation methodology proposed in this section is a generic methodology and can be used for failure traces gathered from any system providing information about the failure events and affected hosts. However, failure traces gathered from Grid5000 system are used in this work. Grid5000 failure dataset was collected for 1.5 years between 2005-2006 and consists of the information of approximately 1300 hosts. The failure traces are downloaded from Failure Trace

Archive (FTA)[88], an online public repository providing failure information gathered from 26 different computing sites. This work has used Grid5000 traces specifically because of the precise details provided regarding the failure start time and end time. The failure dataset consists of the data gathered from 9 geographically distributed computing sites consisting of 15 different clusters. The clusters available in the traces are different from the clusters that are created using cluster analysis techniques. In order to differentiate the representation of trace clusters from the clusters formed after cluster analysis, term 'tclusters' with a notation 'tc' is used for clusters from traces. In this work, only intra-tcluster failure correlation is explored such that failure correlation is present between the nodes belongs to same tcluster in the traces. This is because it is found that nodes belong to same tcluster shares the similar availability (TBF) and unavailability i.e. time to return (TTR) patterns and different from the inter-tcluster nodes (Figure 5.1). Moreover, this assumption is more realistic because nodes belongs to the same tcluster may share the same physical properties, network links and can have more dependence on each other than inter-tcluster nodes [42]. However, inter-tcluster failure correlation is also possible specifically in the case of network failure and power failure. The proposed failure correlation method can also be applied for inter-tcluster failure correlation.

To find the correlated failures in a tcluster tc_a, a node-failure incidence matrix, M_a corresponding to tc_a between the nodes and start time for each failure event is created. In order to create M_a, first the data provided in the failure traces is mined to identify failure event start time information, t_b corresponding to all the failure events for each node, n_k of tc_a. After identifying the start time of failure events, a time space S where $t_b \in S$ is created by including all the failure start time information corresponding to all the nodes of tc_a (Figure 5.2a). After the formation of time space, M_a is created where each cell, cm_{kb} represents an available or unavailable event. Such that if a failure is occurred for a node n_k at time t_b, then cm_{kb} equals to 1 otherwise 0. The whole time space is divided into partitions, P. Each partition is of same size P_{size}, which represents the number of time units covered in a partition. The incidence matrix can further be reduced to a sub-incidence matrix on the basis of the granularity such that the number of time units per partition. Figure 5.2b represents an incidence matrix corresponding to Figure 5.2a with P_{size} equals to 1. By considering more than 1 time unit per partition, the failure correlation can be modelled using both successive

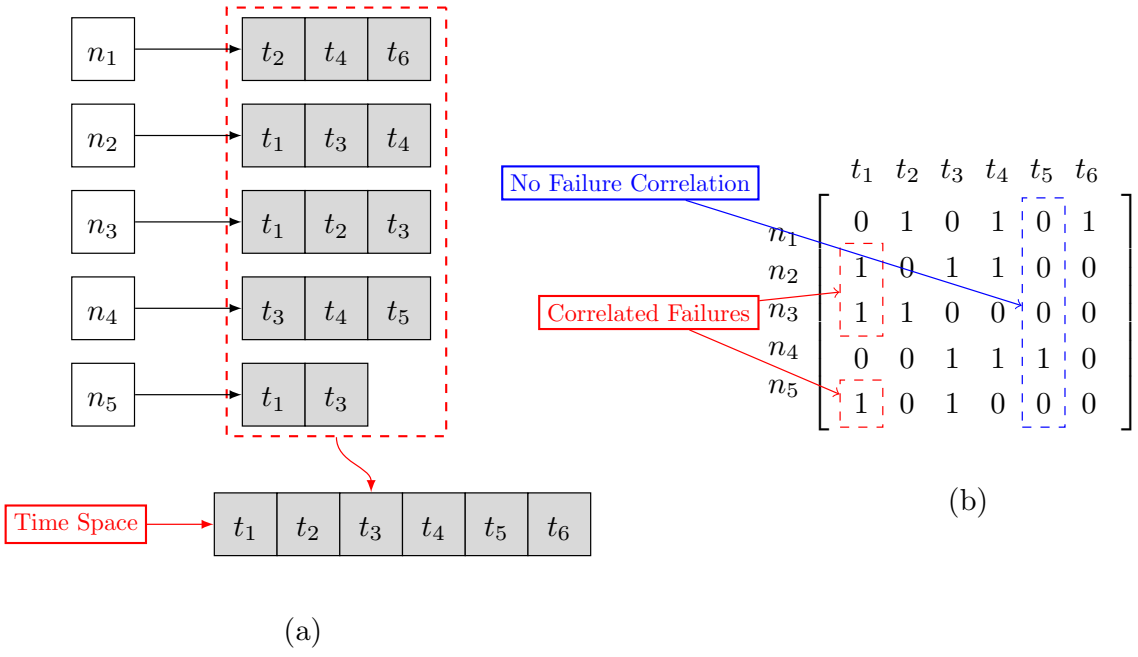


Figure 5.2: (a) Time Space formation by including the start time for each failure corresponding to all the nodes in a trace cluster (b) Node-Failure Incidence Matrix representing the independent and correlated failures

and simultaneous occurrence of failures, which will be explored in future. In order to explore the failure correlation the representation of failure information in M_a is analysed. If two or more failures are occurred at the same time at two or more nodes, then such instance is identified as an event with correlated failures. For example, in Figure 5.2b, failure events occurred at node n_2, n_3 and n_5 at time t_1 are identified as the correlated failures. t_5 is an example of an instance without correlated failures. The same methodology is applied to all the 15 tclusters to create their corresponding incidence matrices and to identify the correlated failures. After modelling the failure correlation, the groups of nodes called clusters with potential of being suffered with a failure at the same time are formed using statistical cluster analysis method. By identifying and grouping of nodes suffered with correlated failures, the occurrence of failures can be avoided by designing correlated failures aware resource management policies. The detailed description about the cluster formation is given in the following section.

5.3.2 Cluster Formation

In order to form clusters both hierarchical cluster and non-hierarchical cluster analysis methods are employed.

$$\begin{array}{c} \\ n_1 \\ n_2 \\ n_3 \\ n_4 \\ n_5 \end{array}\!\!\begin{array}{c} \begin{array}{cccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 \end{array} \\ \left[\begin{array}{cccccc} 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{array}\right] \end{array} \longrightarrow \begin{array}{c} \\ n_1 \\ n_2 \\ n_3 \\ n_4 \\ n_5 \end{array}\!\!\begin{array}{c} \begin{array}{ccccc} n_1 & n_2 & n_3 & n_4 & n_5 \end{array} \\ \left[\begin{array}{ccccc} 0 & 4 & 3 & 4 & 5 \\ & 0 & 3 & 2 & 1 \\ & & 0 & 5 & 2 \\ & & & 0 & 3 \\ & & & & 0 \end{array}\right] \end{array}$$

Figure 5.3: Adjacency Distance Matrix representing the strength of similarity between the nodes by using distance calculated by using equation 5.1. Failure Correlation between two nodes is inversely proportional to the distance between them.

Cluster Formation using Hierarchical Clustering

In hierarchical clustering also know as agglomerative hierarchical clustering, groups are formed by joining smaller groups, iteratively. After identifying correlated failures in an incidence matrix, M_a corresponding to a trace cluster, tc_a, an adjacency distance matrix, A_{c_a} between the nodes is formed (Figure 5.3). The matrix A_{c_a} is used to represent the strength of similarity between the nodes by using distance metrics. The distance between two nodes is calculated as follows

$$d(N_i, N_j) = \sum_{k=1}^{N} \left| a_{ik} - a_{jk} \right|, \quad \text{if } i \neq j \tag{5.1}$$

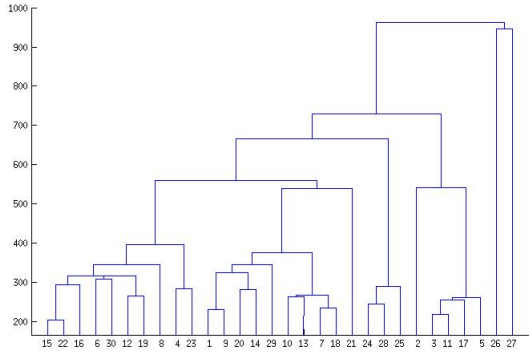
In the given matrix formation method, higher distance between two nodes interpreted as less failure correlation between them and vice versa. The interpretation of distance varies according to the method used for distance calculation. There are metrics other than distance, using which the adjacency matrix can be formed such as Jaccard's similarity coefficient and dissimilarity coefficient. The distance metrics is used to bring the evaluation in accordance to non-hierarchical clustering

algorithm, i.e., K-means algorithm because it uses the distance metrics for cluster formation. By using Equation 5.1, an adjacency distance matrix corresponding to node-failure incidence matrix given in Figure 5.2b is formed (Figure 5.3). The provided matrix is an upper triangular matrix with diagonal elements equal to 0, which represents the distance of a node from itself. After creating the distance matrix, the bottom-up approach is used to form the clusters of nodes using their mutual minimum distances. The given minimum distance calculation method is called single linkage method. The solution provided by hierarchical cluster algorithm is usually visualized using a hierarchical tree knows as dendrogram. Dendrograms (Figure 5.4) corresponding to all the 15 trace clusters are drawn to visualize the cluster formation of nodes with failure correlation.

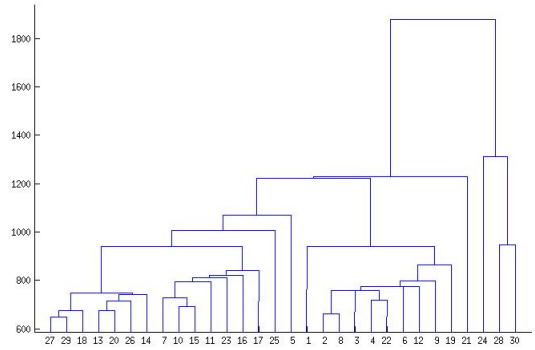
Cluster Formation using Non-Hierarchical Clustering

Use of hierarchical clustering is very advantageous as one can visualize the formation of clusters using dendrogram tree. But the disadvantage of hierarchical clustering is its complexity. It has been argued that hierarchical clustering algorithms are 150 times slower than non-hierarchical clustering algorithms [80] and as the data size increases, application of hierarchical clustering becomes non-feasible. The other disadvantage is that there are no inter-group exchanges in hierarchical clustering, i.e., once an element is assigned to a group, it will never change its group, which some time provides an unoptimized solution. In order to overcome the disadvantages of hierarchical clustering methods and because of large data set, the K-means algorithm is employed to perform non-hierarchical clustering.

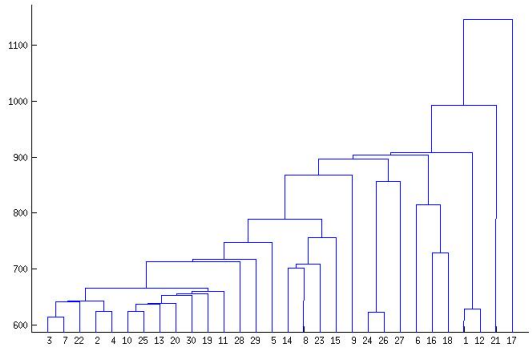
K-means algorithm is started by partitioning the set of nodes, N of a trace cluster, tc_a randomly into K number clusters. After forming the K initial clusters, an iterative operation of inter-partition movement is performed to optimize the placement of nodes in the clusters. The inter-partition or inter-cluster movement for a node, n_k takes place on the basis of the minimum distance from the centroid of each cluster. Such that if a node has a smaller distance from the centroid of any other cluster than the centroid of its current cluster, the node will change its cluster. In order to calculate the distance between the nodes and centroid, mean time between failure (MTBF) corresponding to



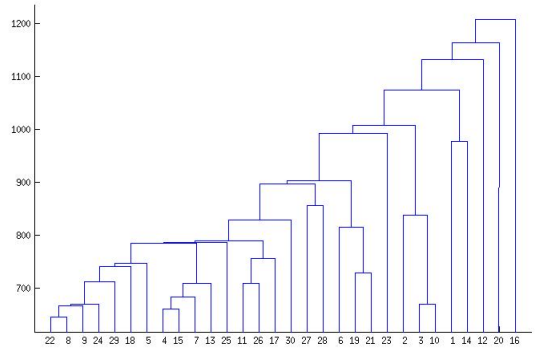
G1/Site1/C1



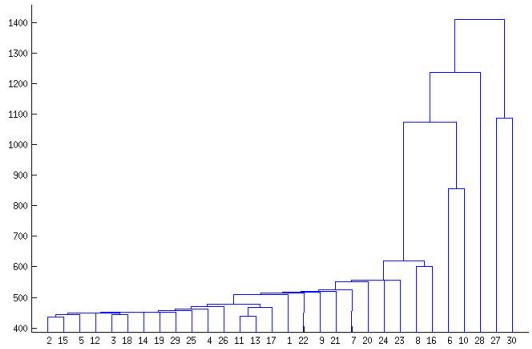
G1/Site1/C2



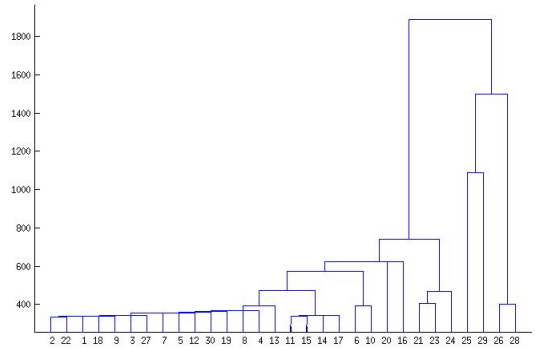
G1/Site1/C3



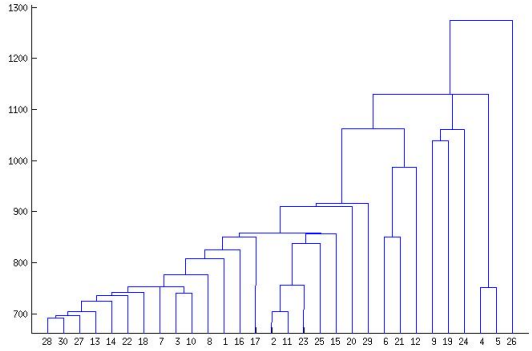
G1/Site1/C4



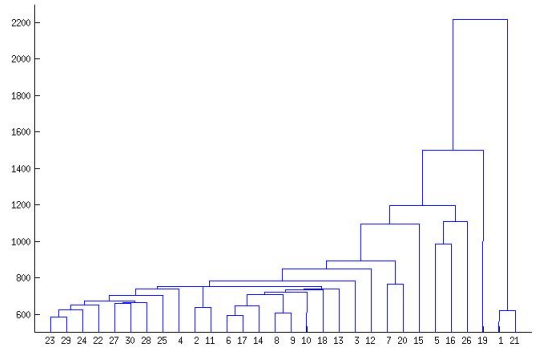
G1/Site2/C1



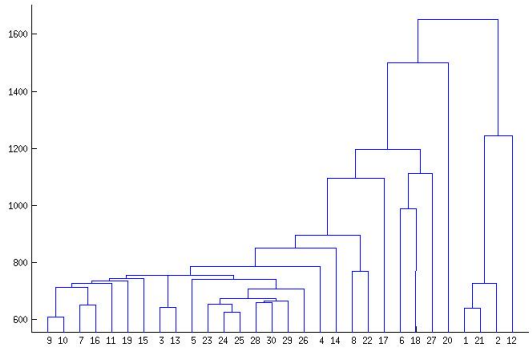
G1/Site3/C1



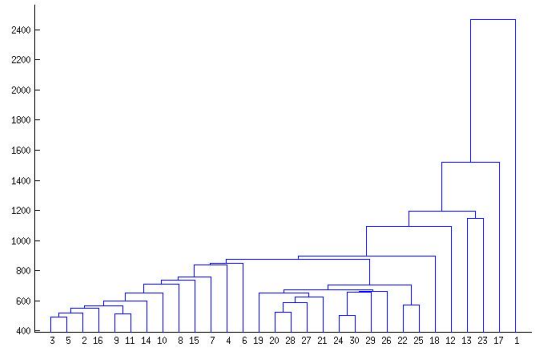
G1/Site4/C1



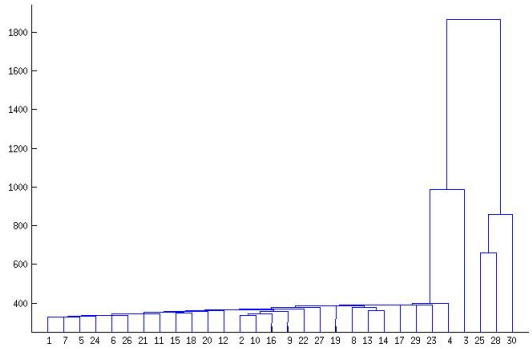
G1/Site4/C2



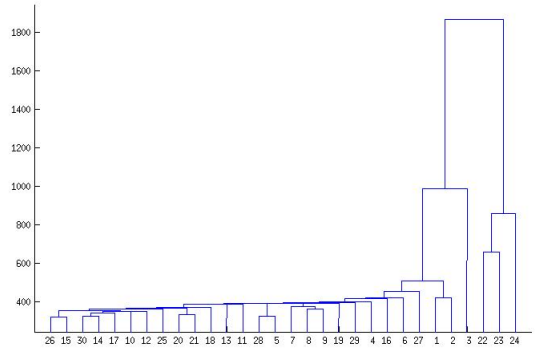
G1/Site5/C1



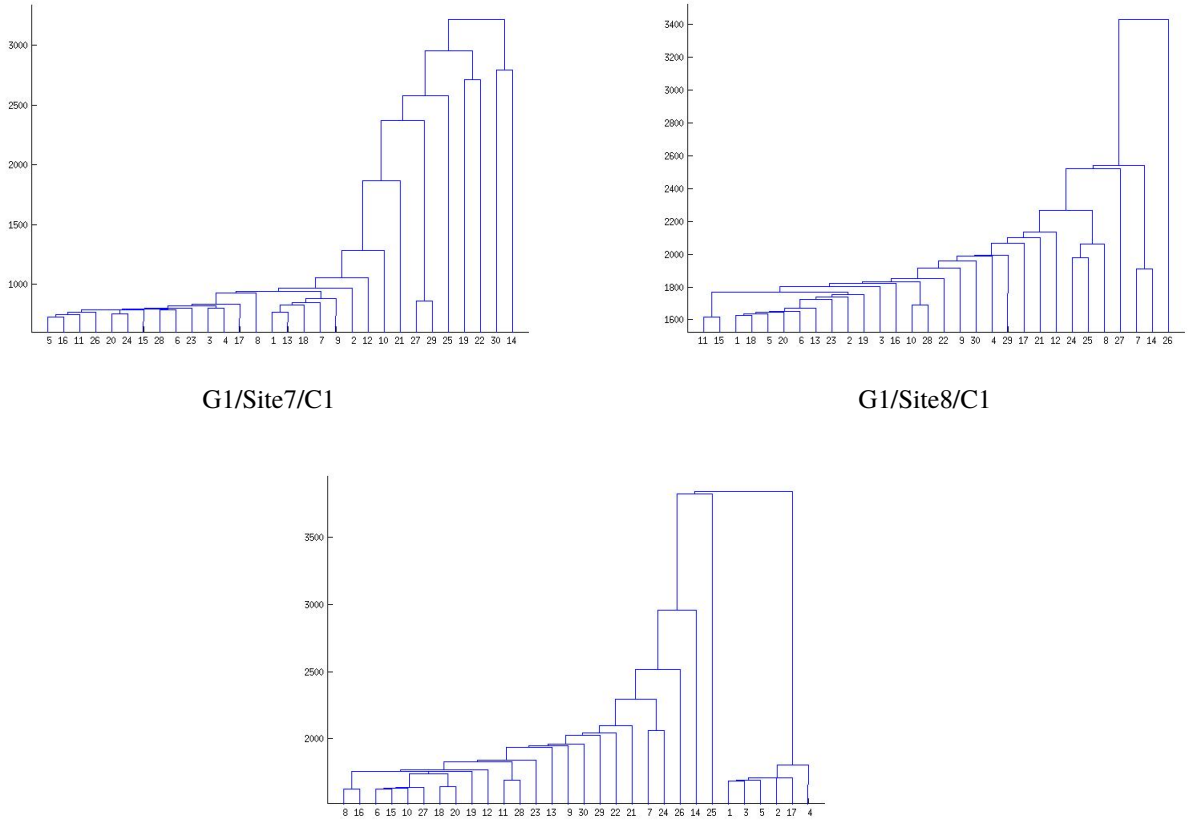
G1/Site5/C2



G1/Site6/C1



G1/Site6/C2



G1/Site7/C1

G1/Site8/C1

G1/Site9/C1

Figure 5.4: Dendrogram for all trace clusters of Grid5000 failure traces. Dendrograms shows the groups of nodes sharing failure correlation at different levels

each node present in the trace cluster, tc_a is used. The centroid of a k^{th} cluster is calculated as an average of MTBFs of all the nodes present in that cluster.

Clustering Criteria

The biggest challenge to perform clustering of a data set is to have an optimality criteria called clustering criteria using which the optimal number of clusters, i.e. value of K can be calculated. Let N represents the set of n nodes in a cluster from traces, tc_a and M represents the set of $MTBF$ for all $n \in N$. If μ_{grand} is the grand mean of set M, then the sample co-variance matrix, V for N is

calculated as follows

$$V = \frac{1}{n} \times \sum_{k=1}^{n} \left(mtbf_k - \mu_{grand}\right) \times \left(mtbf_k - \mu_{grand}\right)^T \tag{5.2}$$

where, $mtbf_k$ is the mean time between failure for node $n_k \in N$ and T stands for transpose. Assume, K number of clusters are formed. Then the mean, μ_p corresponding to cluster K_p consisting of n_p number of nodes is calculated as follows

$$\mu_p = \frac{1}{n_p} \times \sum_{k=1}^{n_p} \left(K_{pk}\right) \times (mtbf_k) \tag{5.3}$$

K_{pk} is the indicator function which represents that a node n_k belongs to K_p cluster. After calculating, μ_{grand}, the scatter of observations, i.e., $mtbf$ corresponding to all the nodes within a cluster, K_p around μ_p by using sum of squares and cross product matrix is calculated. It is also known as pooled within a cluster scattered matrix, S_W and is calculated for all K clusters as follows

$$S_W = \frac{1}{N} \times \sum_{p=1}^{K} \sum_{k=1}^{N} \left(K_{pk}\right) \times \left(mtbf_k - \mu_p\right) \times \left(mtbf_k - \mu_p\right)^T \tag{5.4}$$

After calculating the scatter within the cluster, the value showing the scatter, S_B of cluster means, μ_p around the grand mean, μ_{grand} is calculated by using 'between the clusters sum of squares and cross product matrix' as follows

$$S_B = \sum_{p=1}^{K} \frac{n_p}{N} \times \left(\mu_p - \mu_{grand}\right) \times \left(\mu_p - \mu_{grand}\right)^T \tag{5.5}$$

Once the values for S_W and S_B are calculated then the criteria that is used to check the optimality of the value of K is formalized as

$$O_{val} = min \frac{|S_W|}{|S_W + S_B|} \tag{5.6}$$

By using the value of O_{val} for different values of K, scree plots for all the tclusters are plotted. When the K is equal to 1, the value of O_{val} is maximum and decreases as K increases. When an elbow point appears in the scree plot showing a sharp decline that point is taken as an optimized

value for K. The elbow point represents that the change in the distance after the certain number of clusters is negligible such that shows maximum similarity between the nodes of a cluster. The scree plot for all the trace clusters of Grid5000 failure traces is given in Figure 5.5. From the plots, value of O_{val} is found to be between 5, 6 and 7.

5.3.3 Failure Prediction

As stated in the beginning of this section, a statistical failure prediction method is used to predict the occurrence of failures. Two average based time series analysis methods such as Exponential Smoothing (Section 4.4) and Moving Average are used. The reason for choosing the average based prediction method is the inconsistency and stationarity of the available data. On the basis of the patterns found in the traces, fitting of parametric models such as ARIMA models [114] was tried to predict the occurrence of failures but the Mean Square Error (MSE) was found to be higher than MSE of average based methods. In order to predict the occurrence of failures, prediction of Time between Failures (TBF) is performed. To predict n failures, predictions of $n-1$ TBFs are needed.

After performing the statistical analysis of both average based failure prediction methods, it is identified that with the available data, moving average provides higher failure prediction accuracy by 15% than exponential smoothing. So in the further study, moving average as a failure prediction method is used. Suppose a set of n TBFs corresponding to a node k are observed, $TBF = \{tbf_t \mid 1 \leq t \leq n\}$. The average of t consecutive time series values are taken as the forecasted value corresponding to $(t+1)^{th}$ value (Equation 5.7).

$$(tbf_k)''_{t+1} = \frac{\sum_{i=1}^{t} (tbf_k)'_i}{t}, \text{ if } n > 1 \tag{5.7}$$

This method is also called t-period un-weighted moving average method. Forecasted values for all the $t-1$ values are taken as the simple average of the time series.

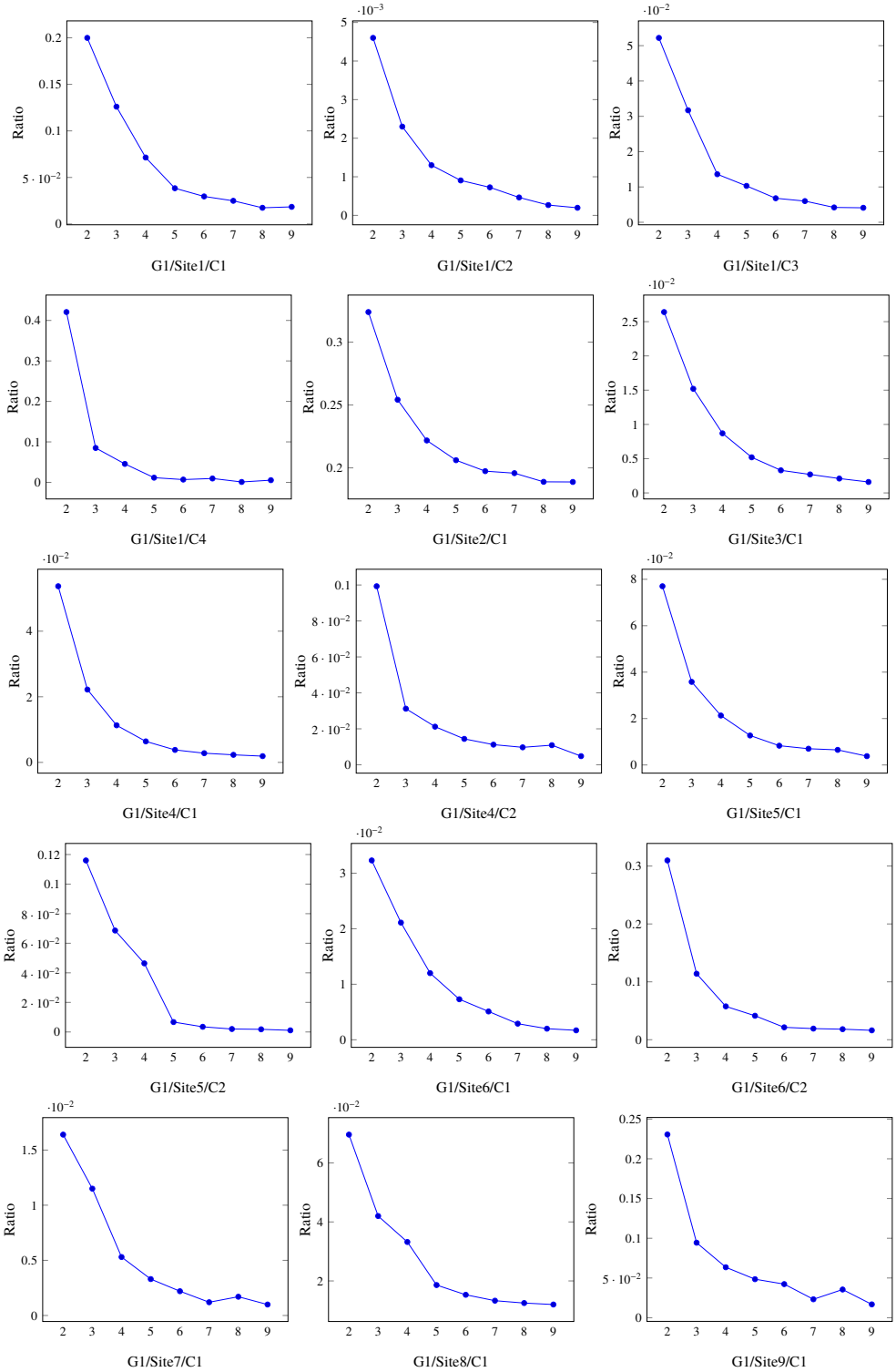


Figure 5.5: Scree Plot for trace clusters of Grid5000 failure traces. Scree plot represents the optimal number of clusters (value of K) by using an elbow point occurs after a sharp decline in the plot.

5.4 Reliability and Energy Modeling

The targeted cloud computing environment (Section 3.3) consisting of a pool, P of failure prone heterogeneous resources/nodes. From the resource pool, resources get provisioned to run the heterogeneous VMs executing a Bag-of-Task (BoT) application in which tasks are arriving at a specific rate.

5.4.1 Reliability Model

The hazard rate, λ_{jk} as a function of utilization, u_k of VM, vm_j while running on a node, n_k is calculated as $\lambda_{jk} = \lambda_{max} \times u_k^{\beta}$ where, β (> 0) is the sensitivity factor towards u_k. When $\beta = 1$, it shows the linear relationship of λ_{jk} with u_k. λ_{jk} is assumed to be following Poisson distribution, such that it remains constant for different levels of u_k and depends on the failure activity of n_k i.e. $\left(\lambda_{max} = \frac{1}{MTBF_k}\right)$. Each node runs multiple VMs and all the VMs running on a node will fail together when the node fails. So the reliability with which n_k will execute m VMs is calculated as follows

$$R_k = \prod_{j=1}^{m} (\exp^{-\lambda_{jk} \times l_i^{max}}) \tag{5.8}$$

where, l_i^{max} is the length of the longest task running on vm_j. While using failure avoidance (FA) or fault tolerance (FT) mechanisms for VMs, extra overheads gets imposed which increases the length of the running tasks and impacts the reliability. In the case of live VM migration as FA mechanism, the migration overhead, MO_{ij} for a task, t_i running on vm_j is calculated as $\left(PMO^{\diamond} + \left(\frac{n \times vm_j^{size} - (P_{size})}{L_{speed}}\right)\right)$, which is the sum of time taken by n pre-copy iterations and pre-migration overheads, $PMO^{\diamond}$. Downtime overhead, DTO_{ij} faced by t_i during migration is calculated as the sum of time taken to migrate the entire copy of vm_j and post-migration overheads, $PMO^{\sim}$, i.e., $\left(\left(\frac{vm_j^{size}}{L_{speed}}\right) + PMO^{\sim}\right)$. So the total VM migration overhead TMO_{ij} is the sum of MO_{ij} and DTO_{ij}. However, when a failure occurs for vm_j, re-execution part, T_i^* and time to return from the failed to running state, TTR_i also adds to the length of a task t_i, besides TMO_{ij} which impacts the

finishing time, F_{ij} of a task.

$$F_{ij} = \begin{cases} l_i + \sum_{p=0}^{n} TMO_{(ij)_p} + \sum_{q=0}^{m} TTR_{(ij)_q} + \sum_{q=0}^{m} T^*_{(ij)_q}, & \text{if } n, m > 0 \\ l_i & \text{otherwise} \end{cases} \tag{5.9}$$

In the case of checkpointing used to provide FT, TMO_{ij} is replaced with checkpointing overheads T''_{ij}. Detailed description about the modelling and formulation is provided in Chapter 4.

5.4.2 Energy Model

On the basis of minimum, P_{min} and maximum, P_{max} power consumption of each node in P, the power consumption for vm_j with utilization u_j running on node n_k is calculated as follows

$$P_k(u_j) = (frac_k \times P_{max_k}) + \left((1 - frac_k) \times P_{max_k} \times u_j\right) \tag{5.10}$$

where, $frac_k$ is the ratio of P_{max_k} and P_{min_k}. As the energy consumption is the amount of power consumed per unit time, the energy consumption by vm_j while executing a task t_i is calculated as the product of power, $P_k(u_j)$ and execution time, l_i of the task. But in the presence of failures, l_i changes because of FA or FT mechanism overheads (Equation 5.9) and occurrence of failures which further impacts the energy consumption. So, the energy consumption by vm_j running on a failure prone node n_k while executing a task t_i of length l_i is calculated as the sum of the energy consumed to execute the actual length of the task and energy wasted to execute the overheads, $E_{vm_{ij}} = \left(P_k\left(u_j\right) \times l_i\right) + E_{waste_{ij}}$. But all the overheads that increase l_i of a task do not contribute to energy wastage. Among the factors such as TMO_{ij}, T''_{ij}, TTR_{ij} and T^*_{ij} that are considered to formulate the task finishing time (Equation 5.9), TTR do not contribute to the energy wastage because during the down-time, a system remains in non-working state. However, the contribution of other factors depends upon the use of either FA or FT mechanism. In the case of VM migration, $E_{waste_{ij}}$ is splitted into two parts, i.e., energy wastage due to TMO_{ij} and T^*_{ij}. As, $TMO_{ij} = MO_{ij} + DTO_{ij}$, MO_{ij} do not contribute to the $E_{waste_{ij}}$ due to the transition state of vm_j and not running on any node. So, $E_{waste_{ij}}$ for VM migration as FA is the sum of $E_{DTO_{ij}}$ and E^*_{ij}.

Despite of a look alike of TTR_{ij}, DTO_{ij} is contributing to $E_{waste_{ij}}$ because during DTO_{ij}, vm_j with task t_i migrates to other node. During this process, an idle resource (core) i.e. resource with 10% or less utilization, u_{idle} get provisioned but not activated. The resource get activated once the VM migration is completed. So, the duration between after provisioning and before activation of a resource with u_{idle} is considered as the contributor to $E_{waste_{ij}}$, which is calculated as follows

$$E_{DTO_{ij}} = \begin{cases} (P_k\,(u_{idle})) \times \sum_{p=0}^{n} DTO(ij)_p, & \text{if } n > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.11}$$

For checkpointing, $E_{waste_{ij}}$ for a vm_j is the sum of energy consumption while saving the checkpoints, $E_{T''_{ij}}$ and re-executing the lost part of a task, E^*_{ij}. The power consumption while saving checkpoints P_{chkpt} is 9 to 11% higher than $P_k\,(u_{idle})$ and much lower than $P_k(u_j)$ [47]. This is because during the creation of checkpoints, the activity of CPU decreases (biggest energy consumer) and activity of I/O controllers i.e. Direct Memory Access (DMA) controller increases in order to perform read/write operations on backup drives. So $E_{T''_{ij}}$ while using checkpointing is calculated as follows

$$E_{T''_{ij}} = \begin{cases} \left(f_{chkpt} \times P_k\,(u_{idle})\right) \times \sum_{p=0}^{n} T''_{(ij)_p}, & \text{if } n > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.12}$$

where, f_{chkpt} is the fraction of idle power consumed during a checkpointing operation. For both VM migration and VM checkpointing, E^*_{ij} due to the occurrence of m failures is calculated as follows

$$E^*_{ij} = \begin{cases} P_k(u_j) \times \sum_{q=0}^{m} T^*_{(ij)_q}, & \text{if } m > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.13}$$

In the absence of any FA and FT mechanisms, only E^*_{ij} will contribute to energy wastage. So the total energy consumption, E_{total} by all provisioned nodes is calculated as the sum of energy consumed by all the allocated VMs, $E_{vm_{ij}}$ while executing their corresponding tasks.

5.5 Resource and VM Management

With the given set of tasks, T and failure prone resources, R, the challenge is to provision the resources and allocate the VMs running the tasks to the resources in order to maximize the reliability and minimize the energy consumption in the presence of correlated failures.

5.5.1 Resource Provisioning and VM Allocation

Before provisioning the physical resources from the pool of resources, P, the RMS first calculate and instantiate the minimum number of VMs (Equation 3.17) and allocate the tasks using bin-packing approach (Algorithm 1). During the creation of VMs, physical resources are provisioned from the pool of resources, P. As the focus of this work is to maximize the reliability and minimize the energy consumption together, Reliability-Energy Aware Best Fit Decreasing (REABFD) policy (Function 3) is used to arrange the resources before provisioning. To avoid or minimize the impact of the occurrence of correlated failures, failure prediction (Equation 5.7) is used. When a failure prediction event occurs for a host belongs to a cluster, it triggers the FA or FT mechanisms for all the hosts in the cluster and marked them as $expected\ to\ fail$. By doing such marking, it is ensured that no new VMs will get allocated to the hosts that are expected to fail in order to reduce the failure losses and overheads. The marking get removed and hosts become available for the allocation to VMs after the occurrence of a failure. Algorithm 6 presents the proposed correlated failure aware resource provisioning and VM allocation policy. Before provisioning a new physical node from P for a VM, first the resource manager checks the availability of free cores on the set of provisioned nodes ($\mathfrak{R}$). If the resource requirement of VM is fulfilled by a provisioned node, then the status of the node is checked such that whether the node is failed or working (step 5). After ensuring the node working, the $expected\ to\ fail$ label is checked (step 8). If the node is labelled as an $expected\ to\ fail$ then a new provisioned node will be searched for. If none of the provisioned node fulfils the resource requirement of VM then a new node is provisioned from P (step 21).

Algorithm 6: Correlated Failure Aware Resource Provisioning

Input: $Pool\ of\ Resources, \mathbf{P}\ and\ List\ of\ VMs, \mathbf{V}$
Output: $Set\ of\ Provisioned\ Resources\ and\ Allocated\ VMs$

```
1  //REABFD Algorithm 3 is called to sort resources of P
   P_sorted ← ReliabilityAndEnergyAware(P)
2  for j ∈ V do
3      VM_cores_j ← vm_j.coresRequired();
4      for k ∈ ℜ do
5          if (RC_k ≥ VM_cores_j) && (S_k ≠ failed) then
6              R ← ρ_k.getClusterResourceList();
7              for t ∈ R do
8                  if (R_t.expectedtoFail() == true) then
9                      flag = true;;
10                     break;
11             if flag == false then
12                 r_k ← vm_j.allocateHost();
13                 RC_k = RC_k − VM_cores_j;
14                 //Calculate VM reliability (Equation 5.8)
                   rel_jk ← vm_j.calculateReliability();
15                 //Estimate VM power (Equation 5.10)
16                 pow_jk ← vm_j.estimatePower();
17                 if (RC_k == 0) then
18                     P_sorted = P_sorted − ℜ_k;
19                 break;
20     //If a new resource needs to be provisioned for VM;
21     if (vm_j.unallocated() == true) then
22         ℜ ← P_sorted;
23         Goto 4
```

5.5.2 Fault Tolerance and VM Consolidation

To mitigate the impact of correlated failures, VM migration, VM checkpointing and their combination (VM migration with checkpointing) are adopted as FA or FT mechanisms. In the case of VM migration (Algorithm 7), on the arrival of a failure prediction event for a node belonging to a cluster, all the VMs running in the cluster get migrated (step 1) in order to avoid the occurrence of failures and algorithm 6 is called to provision new nodes and allocate migrated VMs (step 7). After migrating VMs, all the nodes in the cluster are labelled as $expected\ to\ fail$ (step 9) and label is removed upon the occurrence of a failure for a node. In the case of checkpointing, the present state of VMs is saved as a checkpoint while keeping them running on the same node in order to tolerate the occurrence of failures. In such case, the node will still be labelled as an $expected\ to\ fail$ but it will not available for the allocation to new VMs along with the provisioned nodes belonging to the same cluster. However, while using VM migration and checkpointing in combination, algorithm 7 is used. The only difference is before performing VM migration, the present state of VM get saved as a checkpoint. As the failure prediction mechanisms always subjects to error, there are instances when VMs faced failures. In such cases, VMs get recovered from recent saved checkpoints. Whereas, when VM migration solely worked as a FA mechanism, VM and corresponding tasks get recreated and resubmitted, respectively from the beginning.

After finishing all the allocated tasks, a VM gets terminated. Its corresponding resources sit idle and contribute to energy wastage unless these resources are allocated to a new VM. In order to reduce such idle resources, correlated failure-aware VM consolidation (Algorithm 8) is adopted, which get triggered when a VM is terminated. The proposed method aims to maintain the reliability of the system in the presence of correlated failures while minimizing the energy wastage by considering the reliability profile of a current node (node on which VM is terminated) and target node, which will be chosen to perform consolidation. While performing VM consolidation, new physical resources did not get provisioned and VMs get consolidated between the provisioned physical resources. Before performing VM consolidation, the resource manager needs to decide whether VMs will get consolidated to the current node or out of the current node. This is decided

Algorithm 7: Correlated Failure Aware VM Migration

Input: $Expected\ to\ be\ Failed\ Cluster\ of\ Resources$, **C**
Input: $Pool\ of\ Resources$, **P** $and\ List\ of\ VMs$, **V**
Output: $New\ allocation\ for\ VMs$

```
1 R ← C.resourceList();
2 for k ∈ R do
3     V ← R_k.VmList();
4     for j ∈ V do
5         R_k.deallocate(vm_j);
6         vm_j.setInMigration();
7         //Provision a new host from P for migrating VM (Algorithm 6)
          vmAllocation(vm_j);
8         //Add overheads (Equation 5.9)
9     R_k.setExpectedtobeFailed(true);
```

while selecting a target node. The selection of a target node is made on the basis of three criteria. The first criterion is about ensuring that the target and current nodes do not belong to the same cluster (step 5). By ensuring this, minimization of occurrence of correlated failures is focused on. Second criterion is the number of idle cores. If the number of running VMs on the current node is less than or equal to the idle cores on the target node, then the next failure prediction time of both nodes is considered as the third criterion (step 9). If the failure prediction time of the current node is smaller than the target node then all the running VMs from the current node get consolidated to the target node. However, if the number of running VMs on the target node is less than or equal to the idle cores available on the current node and failure prediction time of the target node is smaller than the current node (step 18), then all the running VMs from the target node get consolidated to the current node. In both cases for current and target nodes, if the node is labelled as $expected\ to\ fail$ then next provisioned node will be chosen for VM consolidation (step 10 and step 19). In case, failure prediction values are not available, hazard-rates of current and target nodes is used.

Algorithm 8: Correlated Failure Aware VM Consolidation

Input: *Provisioned Resources List*, $\mathfrak{R}$ *and Current Host*, **H**
Output: *New host and set of VMs*

```
1  V_h ← H.VmList();
2  C*_h = H.numberofIdleCores();
3  R ← H.getClusterResourceList();
4  for k ∈ ℜ do
5      if (ℜ_k ∈ R) then
6          continue;
7      C*_k = ℜ_k.numberofIdleCores();
8      V_k ← ℜ_k.VmList();
9      if ((C*_k ≥ V_h.size())&&(T~_h < T~_k)) then
10         if (ℜ_k.expectedtoFail() == true) then
11             continue;
12         flag == true;
13         for j ∈ V_h do
14             ℜ_h.deallocate(vm_j);
15             vm_j.setInMigration();
16             vmAllocation(vm_j, ℜ_k);
17             //Add overheads (Equation 5.9)
18     if ((C*_h ≥ V_k.size())&&(T~_k < T~_h)) then
19         if (ℜ_h.expectedtoFail() == true) then
20             continue;
21         for j ∈ V_k do
22             ℜ_k.deallocate(vm_j);
23             vm_j.setInMigration();
24             vmAllocation(vm_j, ℜ_h);
25             //Add overheads (Equation 5.9)
26     if (flag == true) then
27         //Turning off the current host;
28         setState(H, off);
29         break;
30     else
31         //Turning off the target host;
32         setState(ℜ_k, off);
33         break;
```

5.6 Performance Evaluation

After performing a Matlab based verification of all the proposed mathematical models and algorithms, a simulation based validation is performed using a real cloud computing architecture configured with parameters given in Table 4.2. In order to simulate the architecture, well known cloud computing simulator 'CloudSim'[27] is extended by adding failure injector, failure predictor, cluster formation, fault tolerance and VM consolidation.

5.6.1 Simulation Setup

Failures are injected using failure traces downloaded from Failure Trace Archive (FTA)[88]. A brief description about FTA is provided in section 5.3.1. In order to choose the value of window size (w) for moving average method used for failure prediction, a statistical analysis is performed by using failure traces with different window sizes w.r.t the failure prediction accuracy (Figure 5.6). From the analysis, it is observed that as w increases from 2 to 9, the failure prediction accuracy decreases. Consequently, less number of past values contribute to the short term prediction, better prediction results are achieved. This is because of interpolation prediction being performed such that for each failure event value in failure traces, a corresponding failure prediction value is generated. If failure prediction values beyond the number of values provided in failure traces were required to be generated (extrapolation) then the contribution of past prediction values (larger window size) would have been more desirable. In further study, w equals to 2 is used with which the maximum prediction accuracy achieved is between 65% to 76%.

Deadlines corresponding to each task is calculated by using model proposed in Equation 3.1. To keep the deadlines corresponding to tasks moderate [78] and to keep a strict linear relationship [170] between the utilization and reliability, values for stringency factor, f and sensitivity factor, β are set to 1.3 and 1, respectively. Idle power checkpointing fraction, f_{chkpt} is set to 1.15 [47] and other parametric values for VM migration overheads are obtained from [4]. While going in accordance to the configuration of system architecture where one processing core is allocated to one VM which is similar to one of the configurations in Xen hypervisor [26], the considered page

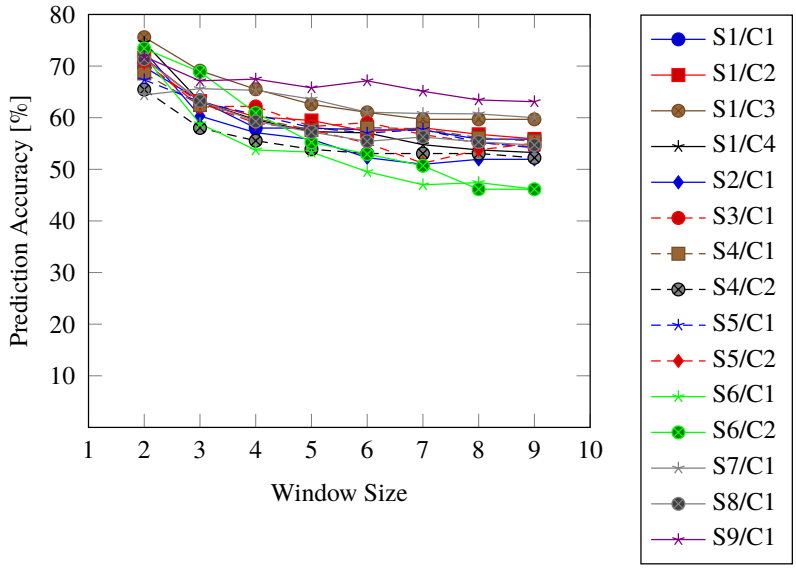


Figure 5.6: Prediction Accuracy vs Window Size. The analysis is carried out by changing the value of t in equation 5.7 from 2 to 9.

size, P_{size} is equal to 4KB which is the default page size for Xen hypervisor. To calculate the power consumption, the values of minimum and maximum power consumption corresponding to a node are taken from spec2008 benchmark[5]. To select the realistic datacenter nodes, the core count and memory capacity of the nodes are mapped with the values provided in the traces. On the basis of mapping information, Intel Platform SE7520AF2 Server Board, HP ProLiant DL380 G5 and Dell PowerEdge R710 as 2, 4 and 8 cores with 4GB, 16GB and 12GB memory nodes are selected, respectively. To generate the BoTs workload, model proposed by Iosup et al. [75] is used with parameters given in Table 3.3.

5.6.2 Results and Discussions

Performance evaluation of the proposed resource management methods using aforementioned FT and FA mechanisms under correlated failures and VM consolidation is done in terms of reliability, task finishing time and energy consumption of cloud computing systems. All the simulations are

[5]https://www.spec.org/power_ssj2008/results/

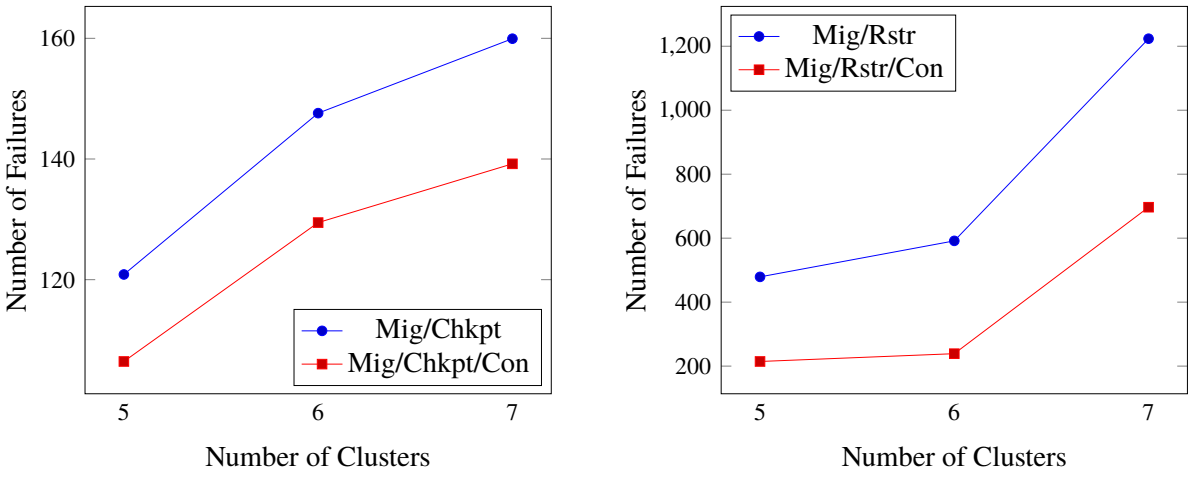


Figure 5.7: Number of Failure vs Number of Clusters

performed using 200 BoTs consisting of total number of tasks between 20000 to 25000. All the reported results are the average of 50 simulations with 95% confidence interval.

For brevity, in the discussion of results below, two environments are using VM migration based mechanisms, i.e., 'Mig' and 'Mig/Con'. In 'Mig', the focus is only to avoid failures by using VM migration. Whereas, in 'Mig/Con', VM consolidation is adopted to regulate the energy consumption dynamically besides VM migration as FA. Both of the environments shows the results of four different scenarios, i.e., 'Rstr', 'Chkpt', 'Rstr/Corr' and 'Chkpt/Corr'. 'Rstr' represents a scenario using VM migration solely as FA without supported by checkpointing as FT. In this scenario, when a failure happens for a node because of failure prediction error, all the VMs and corresponding tasks running on the node get recreated, resubmitted and restarted from the beginning. However, in the case of 'Chkpt' scenario where VM migration is supported by checkpointing, all the VMs and corresponding tasks get recovered from the recently saved checkpoint in case of a failure. 'Rstr/Corr' and 'Chkpt/Corr' represents the scenarios considering failure correlation. In these scenarios, FA and FT is performed at cluster level such that triggering of VM migration or VM checkpointing for one node in a cluster triggers the same for rest of nodes in the cluster. Moreover, in the environment 'Mig/Con', correlated failure-aware VM consolidation is performed for 'Rstr/Corr' and 'Chkpt/Corr' scenarios whereas independent failure-aware VM consolidation is performed for 'Rstr' and 'Chkpt'

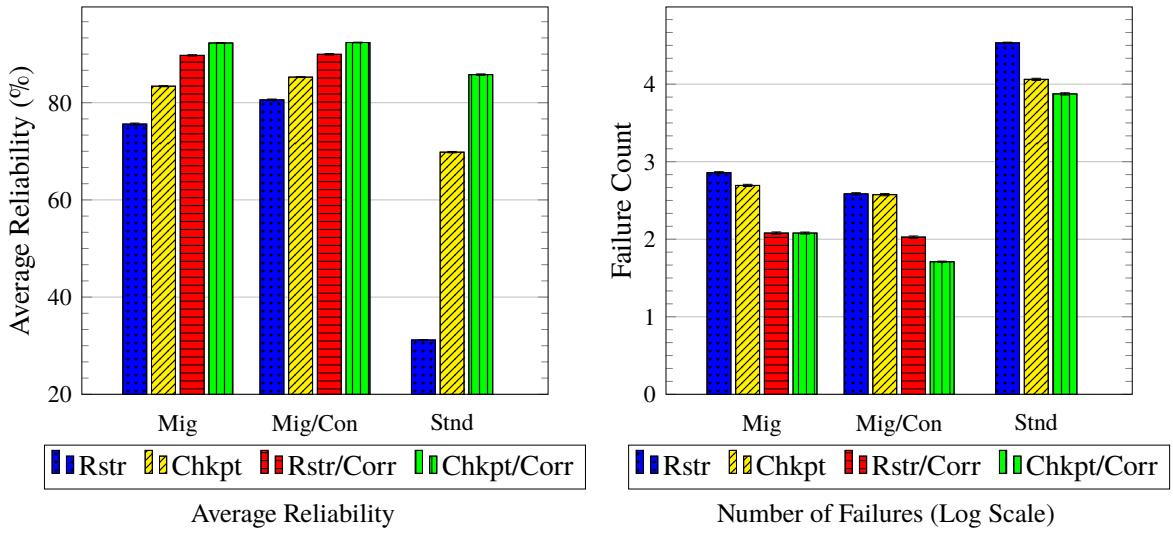


Figure 5.8: Results for Evaluation of Reliability

scenarios. For comparison, a standard environment, 'Stnd' is considered in which all the scenarios i.e., 'Rstr', 'Chkpt' and 'Chkpt/Corr' are neither supported by FA nor VM consolidation. In these scenarios, VMs will keep running or keep getting recreated (in case of failures) on the same physical machines they were allocated to during their creation in the beginning.

On the basis of the statistical results obtained for clustering criteria, O_{val} (Equation 5.6), the optimal number of clusters, K are found to be varied between 5, 6 and 7 for different trace clusters, tc (Figure 5.5). In order to find the most suitable value of K for the current scenario, the simulations using 5, 6 and 7 clusters are performed using all FA and FT mechanisms under failure correlation and VM consolidation. The number of failures occurred is considered as a criteria to decide the value of K. Such that the value of K, which gives the minimum number of failures is considered as the most suitable value and is used for further study. On the basis of results presented in Figure 5.7, it can be seen that K equals to 5 has the minimum failure count while using VM migration (Mig) and Restart (Rstr). In further results and discussions, the reported results are for 5 clusters.

Evaluation of Reliability

Figure 5.8a shows the average reliability obtained by each simulated environment. As obvious, scenario without any FA and FT mechanisms, i.e., 'Rstr' of 'Stnd' shows the minimum reliability. This is because of the highest failure count (Figure 5.8b) and highest turn-around time (Figure 5.9a) such that longer the tasks will run lesser the reliability system will possess (Equation 5.8). With the adoption of FT mechanism (Chkpt) in 'Stnd', a huge improvement of 39% is seen in reliability which is further improved by 16% with the adoption of failure correlation while going in accordance with the reduction of failure count by 15%. As FA mechanism is employed (Mig environment), the average failure count is reduced by 36% for 'Rstr' in comparison to best achieved in 'Stnd' (Chkpt/Corr), which in consequently increased the reliability of the system by 5.78%. This reliability is further increased by 8% by embracing FT (Chkpt) besides FA. Furthermore, by taking failure correlation into account (Rstr/Corr), more reduction of 23% in the occurrence of failures is observed in comparison to scenario with independent failures (Rstr and Chkpt). This reduction improved the reliability of the system by 6% which is further improved by 3% with the adoption of checkpointing as FT (Chkpt/Corr). In the environment using proposed VM consolidation policy (Algorithm 8) in compliance with FA mechanism (Mig/Con), the reliability for scenarios with independent failures (Rstr and Chkpt) is improved by up to 5% with the reduction of 9% in failure count in comparison to their counter parts in 'Mig' environment. However, a negligible change in the reliability is observed for the environments with failure correlation despite of the reduction of approximately, 18% in the failure count. This is because of extra VM migrations took place in order to perform VM consolidation besides the aggressive VM migration for FA. Migrating VMs imposes extra overheads due to which the length of running tasks increases (Equation 5.9) which further reduces the reliability of the system (Equation 5.8). On the contrary, reliability of the system is remained unchanged while performing VM consolidation and achieved impressive energy efficiency results (subsection 5.6.2), which shows the productivity of the proposed correlated failure-aware VM consolidation policy. From the results, it can be said that by considering failure correlation and by putting FA and FT mechanisms together, the occurrence of failures can be

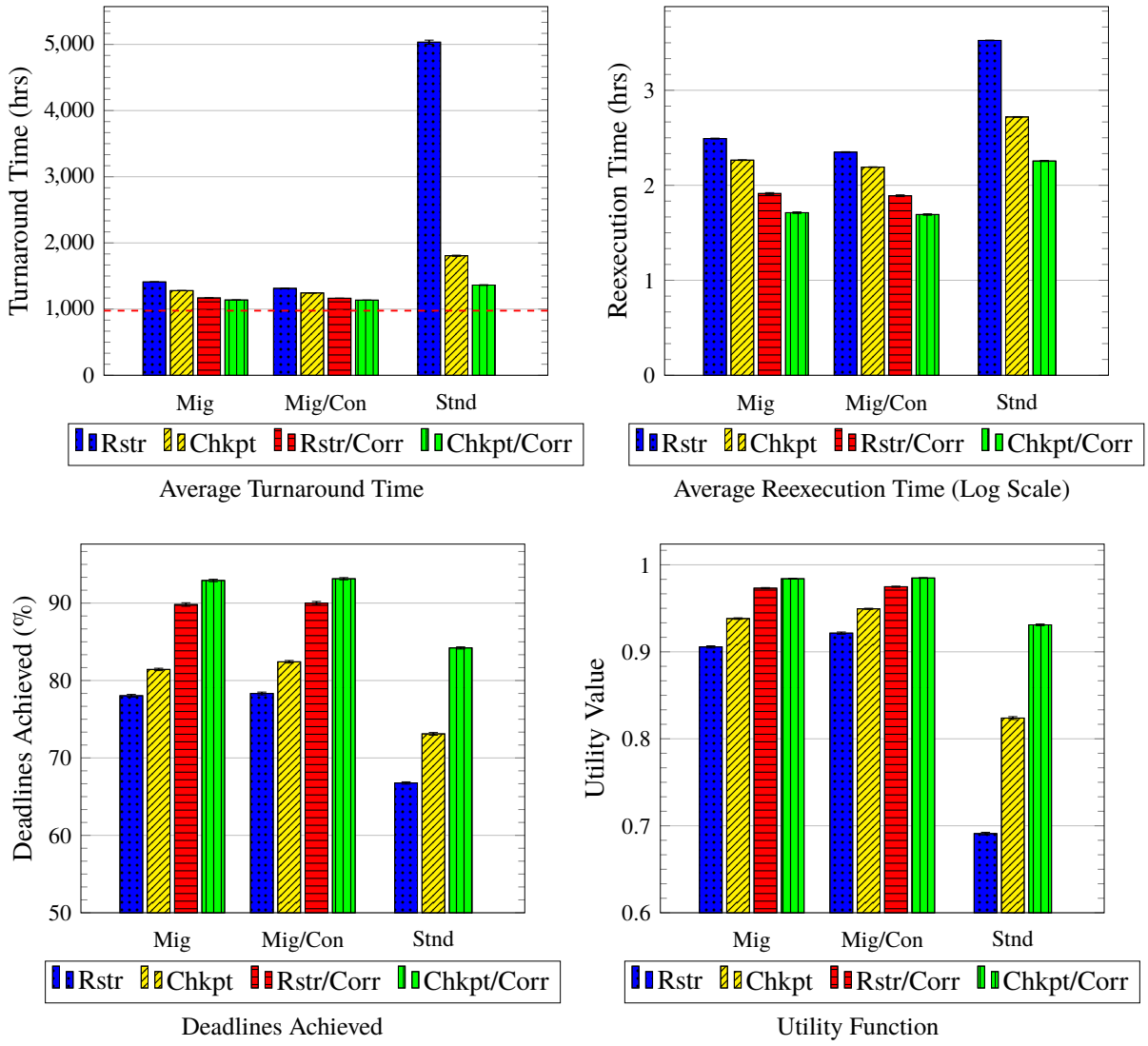


Figure 5.9: Results for Evaluation of Execution Time

reduced and can achieve better reliability of the system in comparison to the environments with independent failures and using both FA and FT, solely.

Evaluation of Execution Time

In order to examine the results corresponding to the energy consumption and energy wastage presented in the following section 5.6.2, it is important to observe the results for task finishing and

re-execution time. Figure 5.9a shows the average turn around time for each task such that the time taken by each task of BoT application to finish. From the figure, it can be seen clearly that 'Rstr' in 'Stnd' environment has the highest turnaround time which is higher by 81% than the turnaround time achieved in a failure free cloud computing environment (represented by using red dashed line). This is because of highest failure count (Figure 5.8b) which resulted in highest re-execution time (Figure 5.9b). But with the adoption of FT mechanism (Chkpt) in the same environment, the average turn around time is decreased more steeply by 64% than failure count. This is because of the recovery of a failed task from the last checkpoint, which causes less re-execution time (lesser by 36%) than restarting from the beginning. With the consideration of correlated failures while supporting the system with 'Chkpt' in 'Stnd' environment, this turn around time is further reduced by 25%.

In the scenario with FA and FT mechanisms (Chkpt in Mig environment) with independent failures, the turnaround time is improved by 6% than what is achieved with failure correlation in 'Stnd' environment. It is further reduced by 11% by considering failure correlation in conjunction with FA and FT mechanisms (Chkpt/Corr of Mig) because of the reduction in re-execution time (Figure 5.9d) and failure count (Figure 5.8b). However, with the adoption of proposed correlated failure-aware VM consolidation policy in the same scenario (Chkpt of Mig/Con), it is expected to have higher turnaround time because of the extra VM migration overheads. But from the results, it is found that the turn around time is remained almost similar to what is achieved in 'Chkpt' scenario of 'Mig' environment. In fact, it is found to be lower by approximately 1%. This is achieved while having a significant improvement in the energy consumption of the system (subsection 5.6.2). This shows the superiority of the proposed correlated failure-aware VM consolidation policy, which is performed without imposing extra overheads.

In order to evaluate the impact of failures on QoS of cloud services while using proposed resource management methods and mechanisms, two metrics, i.e., deadlines achieved (Figure 5.9c and application utility value (Figure 5.9d are adopted. Rather than rejecting a task for a deadline miss (hard deadline), the soft deadline concept is adopted which means it reduces the value (utility value) of the computation for the users. In case of missing a deadline, the utility value for a task

t_i with deadline d_i and completion time c_i is calculated as $\left(\vartheta_i = 1 - \left(\frac{c_i - d_i}{d_i}\right), \text{ if } c_i > d_i\right)$. With the adoption of FT (Chkpt), the achieved deadlines and utility values are improved by 9% and 16%, respectively in comparison to 'Rstr' of 'Stnd' (Figure 5.9d. These values are further improved by 13% and 12%, respectively after the adoption of failure correlation. From all the results analysed so far, it has been observed that while using only FT mechanisms (Stnd environment) if the failure correlation is considered, the achieved performance in terms of reliability and finishing time is similar to the scenarios using FA mechanisms with independent failures despite of high failure count. This is because of less processing overheads imposed by FT mechanisms than FA mechanisms, which helps in reducing the failure count by finishing the tasks earlier. So it can be said that it is not mandatory that only the avoidance or reduction of failures will always gives better results. The overheads imposed by FT and/or FA mechanisms plays a significant role to maintain the system performance. But when the failure correlation is employed in the environments supported by FA mechanisms such as 'Mig', further improvement of 10% and 5% is obtained for achieved deadlines and utility value, respectively. Again adoption of correlated failure-aware VM consolidation (Mig/Con environment) has negligible impact on both metrics despite of extra VM migration overheads.

Evaluation of Energy Consumption

After studying the impact of proposed resource management policies on the system reliability and task finishing time, this section presents the impact of the same on the energy efficiency of the system. The energy efficiency is presented in terms of energy consumption, idle energy consumption and energy wastage. With the adoption of FT mechanism (Chkpt in Stnd), the energy consumption (Figure 5.10a) and energy wastage (Figure 5.10b) is brought down by 49% and 26%, respectively in comparison to 'Rstr' of 'Stnd', which has higher energy consumption by 86% than the scenario without failures (an idle scenario). This improvement is because of lesser turn around (Figure 5.9a) and re-execution time (Figure 5.9b). These values are further reduced significantly by 36% and 20%, respectively with the adoption of failure correlation. In the environment with FA

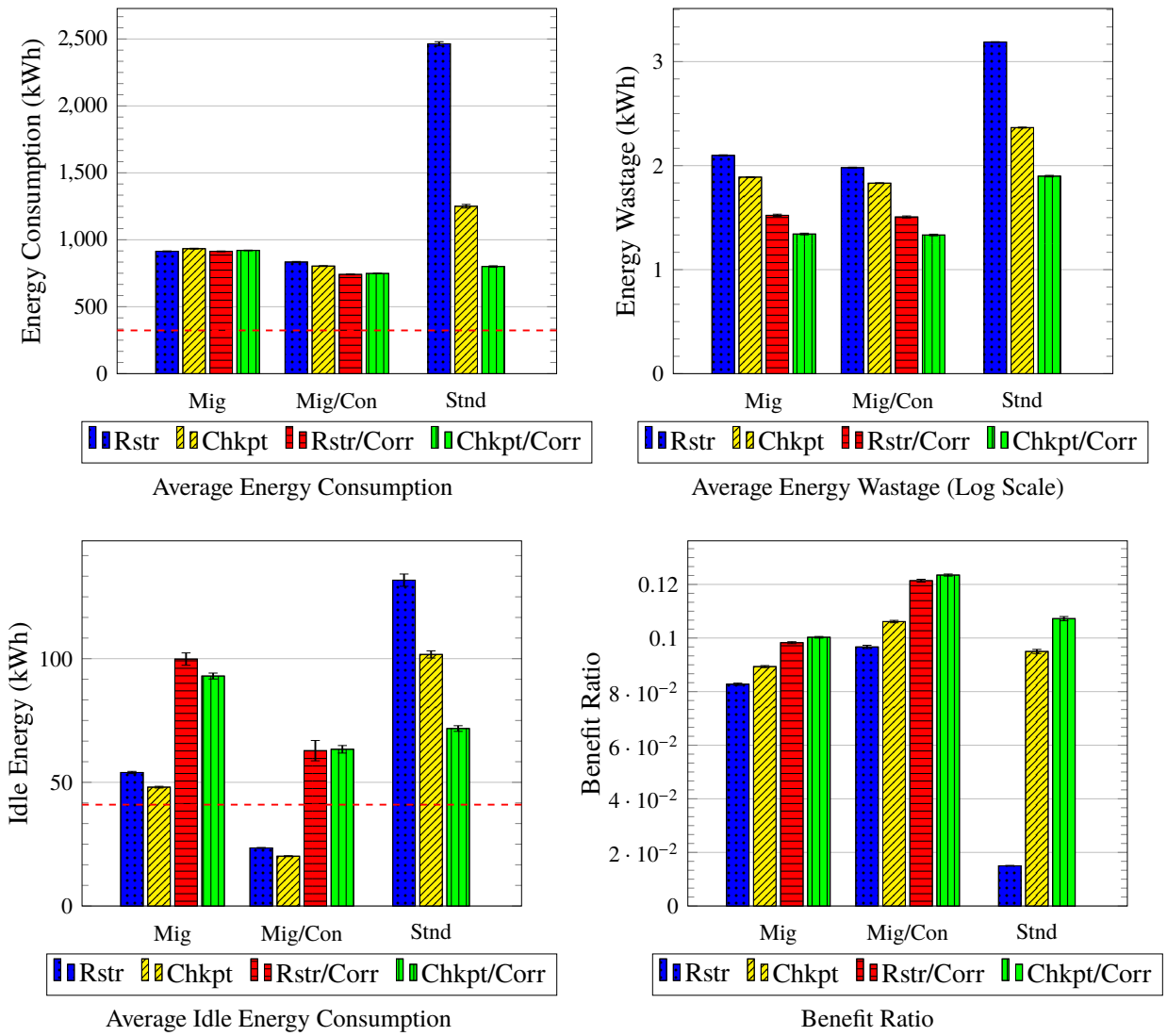


Figure 5.10: Results for Evaluation of Energy Consumption

mechanism (Mig), a strange results are seen. It is found that for all the scenarios either considering independent failures or correlated failures, the energy consumption remained almost same despite of the reduction of around 30% in the energy wastage because of reduction in failure count (Figure 5.8b) and re-execution time (Figure 5.9b). This energy consumption is even found to be higher than what is achieved in correlated failure scenario (Chkpt/Corr) of 'Stnd' environment. The reason behind this behaviour is the provisioning of high number of physical resources by FA mechanisms to accommodate migrating VMs. These VM migrations even get more exaggerated with the adoption of failure correlation where all the VMs running in a cluster get migrated on the basis of a failure prediction event triggered for one host in the cluster. Migration of extra number of VMs led to provisioning of extra physical resources and left high number of provisioned nodes sitting idle. This led to increase the idle energy consumption (Figure 5.10c) by approximately 48% which eventually increased the total energy consumption of the system.

But with the adoption of proposed correlated failure-aware VM consolidation policy (Mig/Con environment), a considerable reduction is seen for idle energy consumption, which is considered as the main metrics to measure the impact of VM consolidation. In the environment 'Mig/Con', while considering independent failures, idle energy efficiency is improved by 51% in comparison to failure free environment. However, Idle energy consumption for correlated failure scenarios of same environment is found to be higher than the scenarios with independent failures because of extra number of provisioned resources to accommodate migrating VMs corresponding to a cluster. However, in comparison to 'Mig' environment, the idle energy consumption is reduced by 37% for Rstr/Corr and 32% for Chkpt/Corr. In general, if only the results of total energy consumption is considered then FT with failure correlation (Chkpt/Corr in Stnd) appears to be outperforming the FA based measures. But as the focus, of this work is to regulate the reliability and energy consumption of the system both at the same time, the ratio of achieved reliability and energy consumption is taken to measure their interplay. This metric is considered as a major metric showing the impact of the proposed mechanisms on the reliability and energy consumption of cloud computing systems (Figure 5.10d).

As expected 'Rstr' of 'Stnd' has the minimum value because it is not supported by any FA and

FT mechanism. The ratio has increased significantly by 86% with the adoption of FT mechanism and Failure correlation. However, it is observed that by considering failure correlation while using only FT mechanisms (Chkpt/Corr of Stnd), better ratio is achieved than what is achieved while using the combination of FA and FT with failure correlation (Mig environment). This happened despite of a good results of 'Mig' in the reduction of failures (Figure 5.8b). The possible reason behind this behaviour is the failure prediction error. Because of the current prediction error, some times VM migration get triggered unnecessarily which imposed extra overheads and resulted in higher energy consumption (Figure 5.10a). By increasing the prediction accuracy, these results probably will change which will be verified in the future work. However, by adoption of proposed correlated failure-aware VM consolidation policy, the benefit ratio is increased in a very promising way specifically for the scenarios with failure correlation. In comparison to the scenario 'Chkpt/Corr' of environment 'Stnd', which is the best so far, the ratio is increased by 13% while using a scenario considering failure correlation supported by both FA and FT mechanisms (Chkpt/Corr of Mig/Con). This shows that even in the presence of current failure prediction error, the proposed correlated failure-aware VM consolidation policy is managed to maintain the equilibrium between the reliability and energy consumption of cloud computing system while achieving the desired quality of services (Figure 5.10d).

5.7 Summary

This work is focused on improving the reliability and energy efficiency of cloud computing systems in the presence of correlated failures. To identify and manage the occurrence of correlated failures, cluster analysis techniques are used. By using the results provided by cluster analysis techniques, reliable and energy efficient resource provisioning and VM allocation policies are proposed. To provide fault tolerance and avoidance, both reactive (checkpointing) and proactive (VM migration) mechanisms are used, which get triggered on the basis of failure prediction results. In order to reduce the energy consumption dynamically, a correlated failure-aware VM consolidation policy is also proposed. Verified by extensive simulation study, following conclusions are drawn

1. By considering correlated occurrence of failures during resource provisioning and VM allocation in cloud computing systems, the service downtime or interruption is reduced significantly by 34% in comparison to the environments with the assumption of independent occurrence of failures.

2. By exploiting failure correlation in conjunction with proposed resource management mechanisms, the turnaround time is reduced by 11% in comparison to the environments using independent failures which in return has improved QoS by reducing the SLA violations and increased the energy efficiency by approximately 20%.

3. The benefit ratio between the reliability and energy consumption of cloud computing systems is improved by 14% by considering failure correlation in association with the proposed failure avoidance and fault tolerance mechanisms.

In the next chapter, we have proposed a reliability-aware cloud computing deployment architecture and explained how the proposed reliability and energy efficient resource management mechanisms can be implemented in the architecture to increase the reliability and energy efficiency of cloud computing systems.

Chapter 6

Framework for Reliable and Energy Efficient Cloud Computing Systems

This chapter proposes a framework to increase the reliability and energy efficiency of cloud computing systems. The framework consists of a cloud computing deployment architecture and its prototype, 'ReliableCloudSim' which is implemented by extending the classes of CloudSim simulator in order to simulate failure-prone cloud computing environment.

6.1 Introduction

In the last decade, the adoption of cloud computing paradigm is increased significantly because of its agility, speed, flexibility and cost efficiency. This trend of adoption became more rapid with the introduction of more data and compute intensive technologies, such as, block chain, Internet of Things (IoT) and big data. As the civilization is set to enter in fourth industrial revolution (Industry 4.0), which will be backed by artificial intelligence (AI), automation and predictive analysis, cloud computing will play a role of bedrock and will act as a critical enabler for the revolution. According to Oracle, cloud computing systems will play a similar role in Industry 4.0 that steam engines played during industrial revolution in late 18th century [34]. Even in these days,

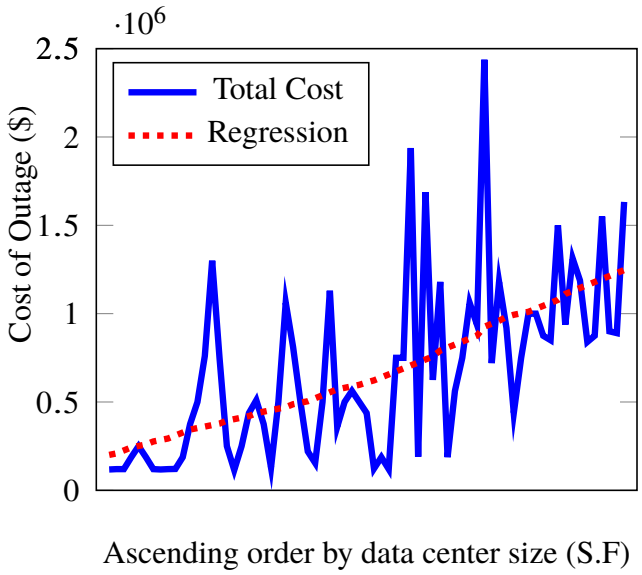


Figure 6.1: Failure cost vs. Data center size [73]

cloud computing underpins many technologies associated to various fields proliferating among users such as, healthcare, education, transportation, banking and industry. According to Mckinsey & company, in 2015, 77% of companies were using traditionally built on-premise IT infrastructure. By the end of 2018, this percentage is predicted to drop to 43% [49]. In a study from more than 1000 organizations ranging from SMEs to large enterprises published by RightScale in 2017 [128], 80% of them are already using cloud services and 14% are planning to adopt it. Such significance and pace of adoption will attract cloud service providers to invest more in the expansion of cloud infrastructure. According to International Data Corporation (IDC), the worldwide spending on public cloud services will reach to $160 billion by the end of 2018, which is higher by 23.2% than 2017. This spending is predicted to reach to $277 billion in 2021 [33].

Such expansion of cloud infrastructure confronts service providers with the challenge of reliability such that ability of cloud providers to deliver services without interruptions or failures. A failure in cloud computing systems cost significantly to both providers and users in terms of business disruption, revenue losses and other post activities performed during recovery and after recovery from a failure. According to a report published by Ponemon institute in 2016, the total cost

of outages in cloud based data centers because of failures is reached to $46,642,491, approximately [73]. These losses because of failures are getting more exaggerated as the cloud infrastructure is expanding to accommodate resource demands. Figure 6.1 shows a linear relationship of the cost of failures with data center size, which is the result of legacy deployment methods cloud providers are using. Industry experts have acknowledged the fact that cloud computing technology and applications are changing rapidly but approaches for cloud architecture and deployment could not keep pace with the change. In short, improving only technology and increasing size and number of data centers without focusing on architecture and deployment models can not prevent downtimes in data centers. This is illustrated in October 2013, when the whole Microsoft Azure system went down while updating a Red Dog Front End (RDFE) module. This incident happened because of the way Azure is architectured and deployed. The developers could not run one instance of RDFE in one part of the system and then deploy it to rest of the system after it had proven working properly. Instead, the update had to be performed on the whole Azure system centrally, which exposed the existing bug and triggered an outage across all the regions of Azure [43].

Due to the revenue losses incurred because of failures, service providers are opting for cyber insurance. But the challenge for both insurance providers and service providers is to estimate the potential business losses and the amount needs to be covered by insurance. As in chapter 5 shown, the occurrence of a failure can trigger series of failure events in a correlated manner which can disrupt the web-based businesses, payment orders or other systems supported by clouds. Such occurrence of correlated failures can result in huge losses which can be easily under-estimated and under-insured. According to an emerging risk report published in 2018 by Lloyd and AIR Worldwide [3], a failure with a duration of 3 to 6 days in a top-tier cloud computing system could result in losses of between $5.3 billion to $19 billion. Out of these losses, only a fraction such as between $1.1 billion to $3.5 billion will be insured, which varies on the type of application running on a cloud.

In order to mitigate the impact of a failure, cloud computing providers must have a deployment framework which will take the reliability factors into account besides other operational attributes. The prototype realization of such deployment framework is also required before the actual imple-

mentation. Such practice will avoid the situations similar to Microsoft Azure's global failure [43] such that any application will be deployed and tested on the prototype framework before making it live. The prototype realization of framework will also help to estimate the impact of failures on revenue losses and the operational expenses incurred with the adoption of fault tolerance or failure mitigation methods. From the perspective of cyber insurance providers, the framework and its prototype will help them to identify and understand the nature of the risks. Such understanding will help the insurance providers to model the revenue losses incurred because of a failure. This will further help the insurance providers to set-up the guidelines which will be helpful to reduce the gap between the actual losses and insured losses. This will assist the service providers and users to prepare for and recover from the extreme scenarios of failures. Such practice will help both service providers and cyber insurance providers to grow their business in a controlled and prudent manner.

In order to fulfil the reliability requirements of cloud service providers, estimation requirements of cyber insurance providers and quality of service requirements of cloud service users, this chapter is proposing a reliability-aware cloud computing deployment architecture. The proposed architecture considers the failure characteristics of the physical resources while taking any resource management decisions. A Java based prototype of the architecture is also proposed, and it can be used by cloud providers and insurance providers to carry out research to measure the impact of the occurrence of failures on the running applications and corresponding losses. This will help them to define new resource management policies and service level agreements, which will provide better reputation to their products and will help them to expand their businesses.

6.2 Reliability and Energy Management Cloud computing Architecture

Figure 6.2 shows the cloud computing architecture focusing on reliability attributes. Architecture is representing a cloud computing system consisting of a pool of failure prone heterogeneous resources/nodes either active or idle. From the resource pool, resources get provisioned to run

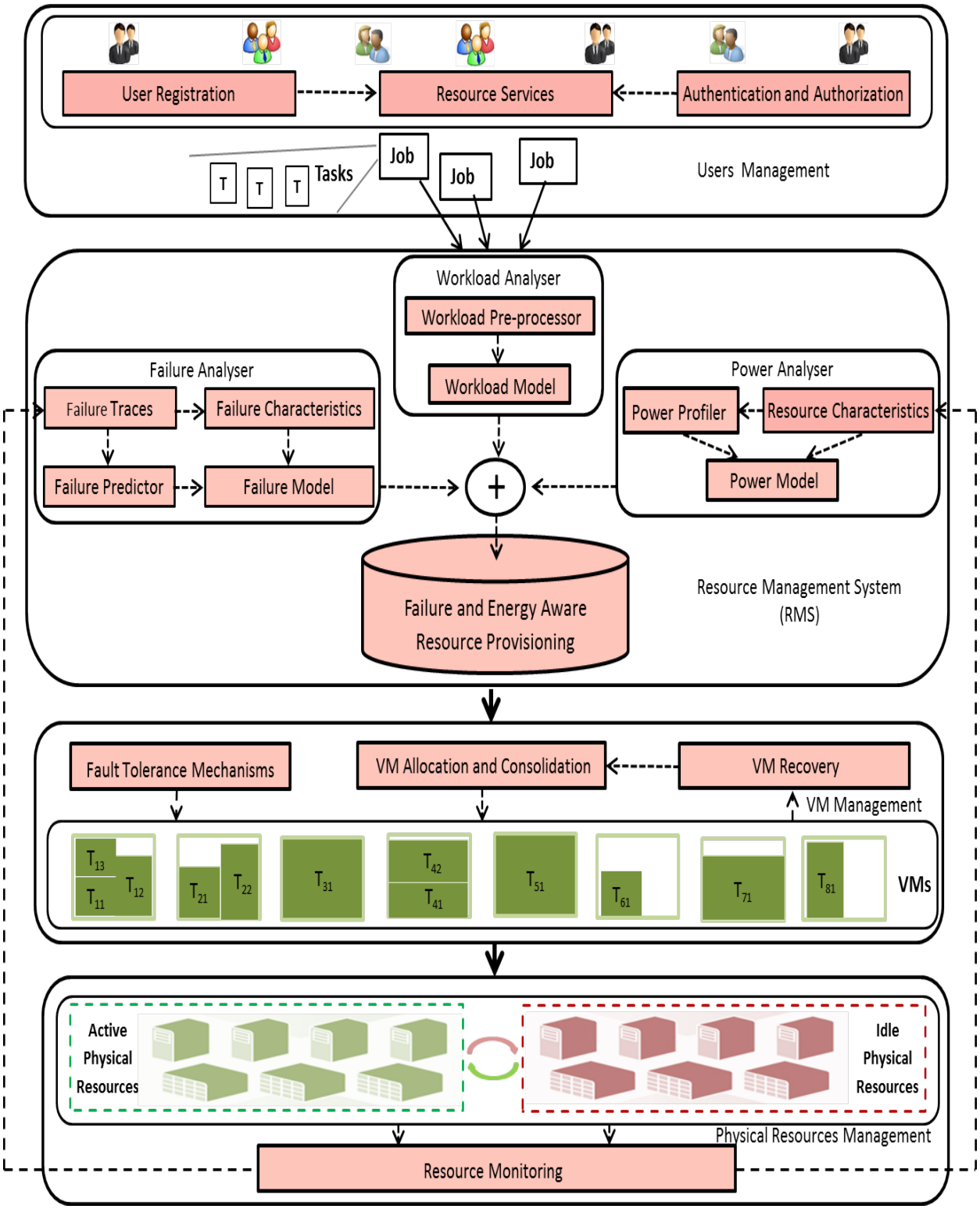


Figure 6.2: Reliability-Aware Cloud Computing Deployment Architecture

the heterogeneous VMs executing different kind of workloads or applications. For the clear representation, the architecture is organized in layers representing all the core functions from users to infrastructure. The core functions of each layer are described as follows.

6.2.1 Users Management

Users management system provides an abstract view of the system by hiding all the internal components of the architecture. It provides an interface similar to AWS, using which users can configure and avail the services of underlying system.

User Registration

Before accessing the system, users need to register themselves. The user details will be stored in a user database and will be used by authentication and authorization module to verify the authenticity of the users.

Authentication and Authorization

This service performs authentication and authorization, which allows the users to login and access the services of the system.

Resource Services

This is the main module in users management layer, which provides information about different types of available services, resources and corresponding attributes such as resource location, cost, etc. It provides web interfaces using which users can perform resource reservation and configuration according to the requirements of their applications. After performing all such operations, users management layer let the users to submit their workloads to the underlying cloud infrastructure for execution.

6.2.2 Resource Management System

Resource management system (RMS) is the heart of the architecture, where all the core services such as reliability management, energy management and workload management are existing. On the basis of the outputs of core services, RMS takes the decision about resource provisioning and scheduling. The main focus of this architecture is to deploy the cloud computing systems in a reliability manner in order to reduce the losses and operational expenses because of the occurrence of failures but other metrics are required to measure the impact of the decisions made by the policies and mechanisms incorporated in the architecture. This thesis focus on energy efficiency of cloud computing systems as other metrics besides reliability. RMS has an incorporated power analyser, which is used to regulate the energy consumption of the system in the presence of failures.

Workload Analyser

Workload analyser is like a presentation layer in OSI reference model for Internet, which brings the data in a required and acceptable format. It consists of workload pre-processor, which pre-process the workload submitted by upper layer according to the workload models provided by workload model module. Workload model module is responsible for defining the format of data accepted by the system. Identifying the task inter-dependency in the case of workflows, defining task deadlines are the operations performed by workload analyser. It also calculate the number of VMs required to create and initiate in order to execute the submitted workload. Service of creation and allocation of VMs to physical resources is provided by VM management unit of the architecture.

Failure Analyser

This module of RMS is responsible of monitoring the failure activity of physical resources, which is performed by using real time logs/traces gathered from the physical resource management unit of the architecture. Failure prediction module is also managed by the failure analyser, which predicts the occurrence of failures by performing certain statistical operations by using gathered failure traces. Failure prediction results are further fed to failure model. Failure model also takes input

from failure characteristics module which identifies the attributes corresponding to each failure event such as type of failure, reason of a failure, start and stop time of a failure and correlation with previous or successive failures. By considering input from both failure prediction and failure characteristic modules, failure model provides input to resource provisioning algorithms. Bu using the input, hazard rate of resources and reliability of workload is calculated and decisions about reliability aware resource provisioning are taken.

Power Analyser

To regulate the energy consumption, power analyser is employed which considers the resource characteristics such as hardware configuration of physical resources to create their power profiles such that power consumption on different levels of utilization of a system. Power model brings the input of resource characteristics and power profiler together and provides input to resource provisioning unit to perform energy efficient resource provisioning.

By taking inputs from workload, failure and power analysers, resource provisioning unit provisions the physical resources in a reliability-aware and energy-aware manner. All the algorithms and policies proposed in chapter 3, 4 and 5 are incorporated in Failure and Energy Aware Resource Provisioning module of RMS.

6.2.3 VM Management

The VM management unit is responsible for the creation and allocation of VMs to the provisioned physical resources by RMS. This unit consists of all the fault tolerance mechanisms adopted to provide fault tolerance to the running VMs in order to finish the submitted workload without violating the deadlines. In an instance of occurrence of a failure, recovery of failed VMs and corresponding tasks are also handled by this unit.

VM Allocation and Consolidation

This module is responsible for the creation and allocation of VMs according to the input received from workload analyser about the number of VMs required to be created to execute the workload submitted by users. Once the VMs are created, the unit allocates the VMs to the provisioned physical resources acquired by RMS. Algorithms 1, 3 and 6 for the creation and allocation of VMs are incorporated in this module of VM management unit. Decisions about VM consolidation to reduce the energy consumption dynamically is also taken by this unit. In order to perform VM consolidation in the presence of independent and correlated failures, this unit is integrated with algorithms 5 and 8, respectively.

VM Recovery

When a VM failure happens, all the corresponding tasks get failed. In order to recover from a failure, VM recovery module takes the necessary steps such as resubmission of failed workload, recovery of a VM state from the last state, if any fault tolerance mechanism is employed. In the absence of fault tolerance mechanism, VMs will be recreated and execution of corresponding workload will restart from the beginning.

Fault Tolerance Mechanisms

This part of VM management unit plays a very crucial role by providing fault tolerance to running VMs to provide uninterruptible service to users. Different reactive and proactive fault tolerance and avoidance mechanisms such as VM checkpointing and VM migration, respectively resides in this part of the architecture. These mechanisms can get triggered periodically (Section 3.4.1) or based on the failure prediction results provided by failure analyser of RMS.

6.2.4 Physical Resources Management

The bottom layer is the physical resource management layer where all the decisions about the activation and deactivation of physical resources are taken on the basis of the input provided by

upper layers.

Resource Monitoring

This unit is responsible of monitoring the activity and status of physical resources such that occurrence of failures, hardware configuration and power profiles and gathers the real time data. All the monitored data is then fed to the failure analyser and power analyser of RMS. On the basis of the outputs of failure and power analyser and workload characteristics submitted by users, RMS takes the decision about failure and energy aware resource provisioning and VMs get allocated to the provisioned physical resources to execute the applications of users.

6.3 Implementation and Prototyping

To implement the prototype of proposed failure-aware cloud computing deployment architecture, an environment is required which provides all the functions required to set-up units of the proposed cloud computing architecture. The environment will enable to set-up a test bed for the evaluation of new methods and policies in a failure prone cloud computing environment. In this respect, ReliableCloudSim is developed to provide support for modeling and simulation of failure prone cloud computing systems. ReliableCloudSim environment is built on top of CloudSim simulator [27], which is currently the most sophisticated and accepted discrete event simulator for cloud computing systems.

6.3.1 Simulator Architecture

Figure 6.3 shows components of ReliableCloudSim (highlighted in grey) integrated with multilayer CloudSim architecture. In the following section, detailed description about new components corresponding to ReliableCloudSim is provided besides brief explanation of CloudSim components. Detailed description about CloudSim architecture can be obtained from [27].

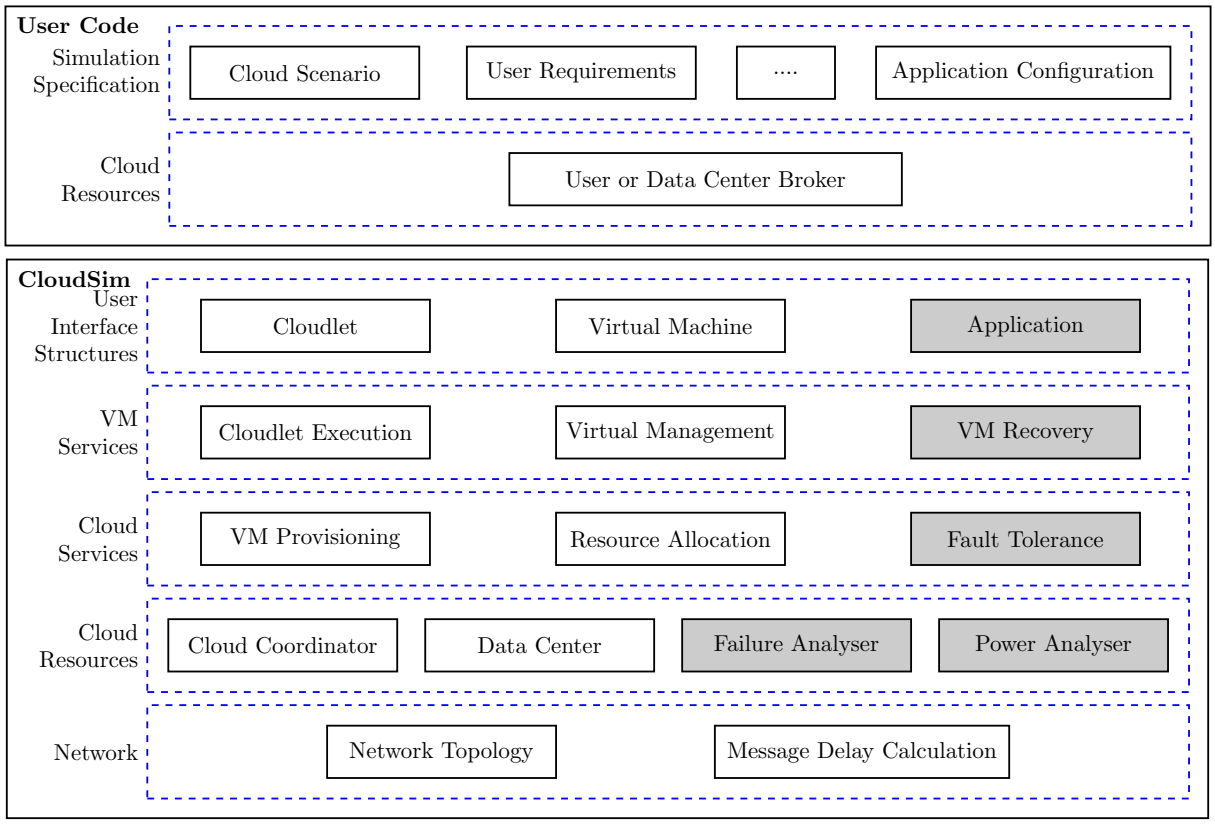


Figure 6.3: ReliableCloudSim Simulator Architecture showing the integration of components of ReliableCloudSim (highlighted in grey) with the standard CloudSim Simulator components.

Application

This component is a part of User Interface Structure layer of the simulator. It performs the workload profiling after performing analysis of workload submitted by users or data center broker and brings it in a right format to make it executable on underlying cloud infrastructure. This component also provide input to other components of the same layer such as Virtual Machine by using which the system calculate the number of VMs required to provision or create in order to execute the submitted workload.

VM Recovery

This component is a part of VM Services layer of the simulator. VM Services layer is responsible for managing the running VMs and execution of workload (cloudlets) submitted to them. This layer keeps track of executing and finished cloudlets and provides input to upper layer about the available capacity on the running VMs in order to accommodate new cloudlets. This layer also provides input about the current utilization of running VMs to lower layers required to evaluate the reliability and power. VM recovery module provides the services required to recover a VM and corresponding cloudlets to their last working state in the case of a failure. It is responsible for providing fault tolerance at VM level. In the case of checkpointing, this module provides the service to checkpoint all the running cloudlets corresponding to a VM. For VM migration, it provides service to migrate running VMs and corresponding cloudlets to safe and reliable physical resources in order to tolerate and avoid the occurrence of failures.

Fault Tolerance

This module is responsible for supporting cloud services with fault tolerance in order to provide an uninterruptible service at user end. The overall responsibility of Cloud Services layer is to manage the process of creating the VMs requested by upper layers and allocate them to physical resources satisfying their resource requirements. Besides this, cloud services are supported with different type of fault tolerance mechanisms such as VM migration, VM checkpointing and VM replication, which are flexible to be selected. The selection of fault tolerance mechanisms either depends upon the user or service provider requirements. Fault tolerance module provides service to upper layer to recover VMs from a failure.

Failure and Power Analyser

These modules provide services corresponding to failure analyser and power analyser components of cloud computing deployment architecture proposed is Figure 6.2. While working with cloud co-ordinator and data center components responsible of managing activities of physical cloud

infrastructure, failure and power analyser gathers the real time data corresponding to resource activities such as resource utilization, failure events and energy consumption. By using real time data, failure prediction is performed. On the basis of failure prediction results, these modules provide services as an input to fault tolerance module of upper layer, which further triggers the fault tolerance mechanisms in order to provide uninterruptible cloud services to the users.

6.3.2 Design and Implementation

As stated earlier, ReliableCloudSim simulator is developed by extending the CloudSim simulator. The new simulator enables the users to simulate the behaviour of a failure prone cloud computing system. There are four main classes which regulates this behaviour, such as, FailureDatacenter, FailureDatacenterBroker, FailureGenerator and PowerModel. A UML class diagram for ReliableCloudSim is shown in Figure 6.4. This section describes the classes corresponding to ReliableCloudSim.

FailureDatacenter

This class is an extension of Datacenter class of CloudSim. This is a core class implementing the functions of Cloud Resources layer of simulator architecture (Figure 6.3). By using this class, homogeneous and heterogeneous physical resources forming cloud based datacenter is defined. The simulation of occurrence of failures injected by using failure generator is performed by using the functions of this class. Functions corresponding to all the fault tolerance mechanisms used in this thesis (Chapter 3, 4 and 5) are the part of this class as well. This class also consists of function defining VM consolidation used in chapter 4 and 5.

FailureDatacenterBroker

This class is an extension of DatacenterBroker class of CloudSim. This class simulates the behaviour of cloudlet failure occurred because of VM failures triggered by FailureDatacenter class. Besides cloudlet failure simulation, this class also performs basic functions of DatacenterBroker

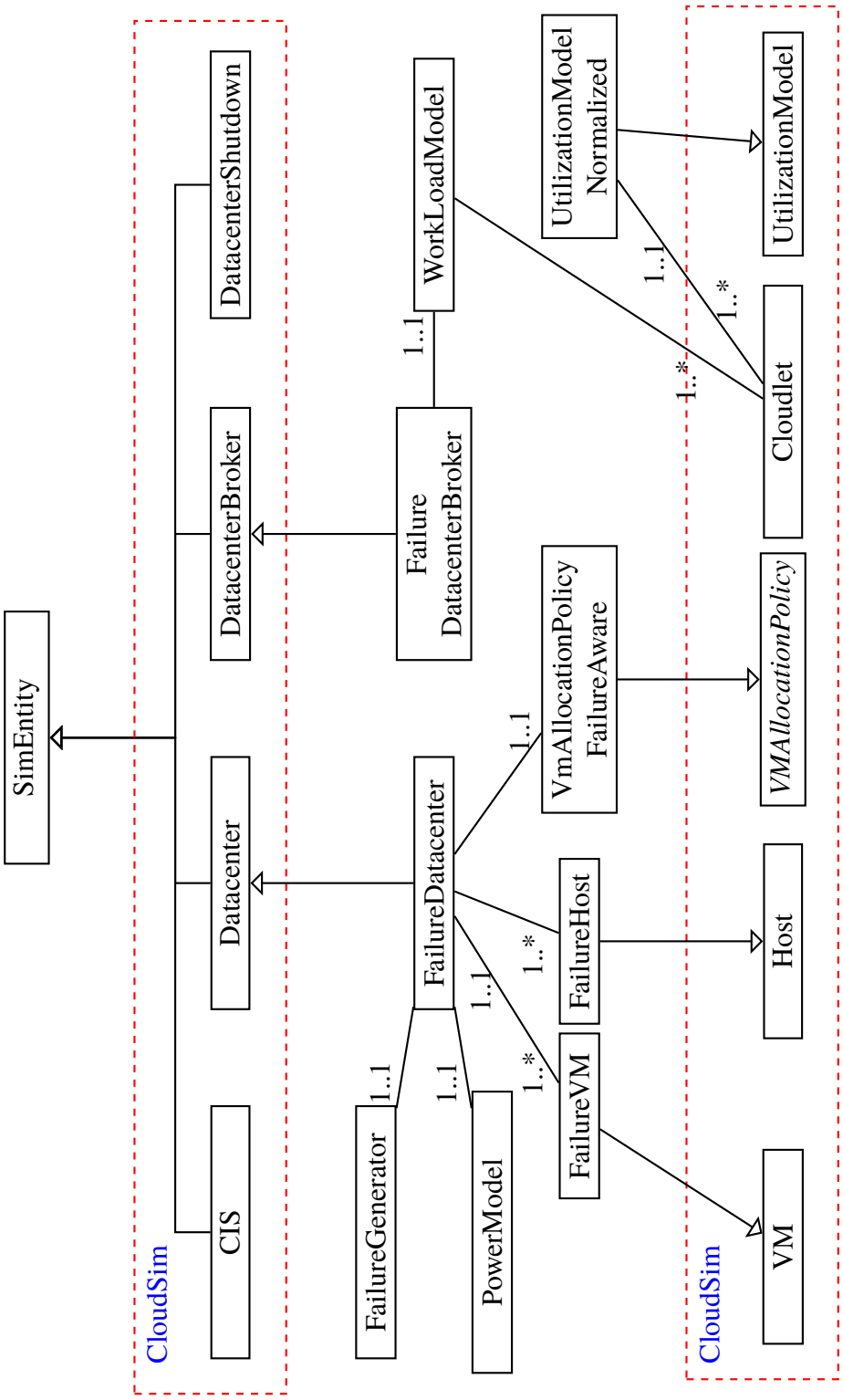


Figure 6.4: Class diagram of ReliableCloudSim representing the interaction between the classes of ReliabileCloudSim and standard CloudSim simulator

class defined in CloudSim such as negotations with cloud coordinator to get optimized resource allocation, creation of VMs in data center and allocation of cloudlets to VMs.

FailureVM

This class is an extension of class VM and simulate the behaviour of a failure prone virtual machine.

FailureHost

This class is a subclass of Host class of CloudSim and simulate the behaviour of a failure prone physical server. Beginning with an arrival of a failure event, this class triggers the object of FailureVM class which further calls FailureDatacenterBroker to trigger failure among all the running cloudlets and corresponding VMs mounted on the server.

FailureGenerator

This class reads the real failure traces gathered from Failure Trace Archive [88] and injects the failures using start and stop time corresponding to failure events available in the traces. Hardware configuration and failure characteristics corresponding to all the physical resources present in FailureDatacenter class are obtained from FailureGenerator class in order to calculate the power and reliability of the system.

PowerModel

On the basis of hardware configuration of resources provided by FailureGenerator class, object of this class is used by FailureDatacenter class to define the power profile for each physical resource of data center. The power profiles are further used to calculate the energy consumption and impact of failures on energy consumption during run-time.

WorkLoadModel

WorkLoadModel class is used to define the behaviour of an application, users want to execute on cloud infrastructure. It defines all the attributes corresponding to an application such as task/cloudlet length, data rate and task dependency or in-dependency. WorkLoadModel provides input to FailureDatacenterBroker class, which calculate the number of required VMs to accommodate the cloudlets or tasks and initiates the VMs in FailureDataCenter class and allocate tasks to VMs using task allocation algorithms defined in previous chapters.

VmAllocationPolicyFailureAware

This class is an extension of an abstract class, *VmAllocationPolicy* defined in CloudSim simulator. This class consists of all the crucial functions required to provision the physical resources and allocate them to VMs during VM creation, migration and consolidation phases. The object of this class is defined in FailureDatacenter. All the resource provisioning and VM allocation policies proposed in this thesis are the part of this class.

UtilizationModelNormalized

This class is an extension of UtilizationModel class of CloudSim. This class consists of functions required to calculate the utilization corresponding to each cloudlet by normalizing w.r.t longest cloudlet. The normalized utilization further used to calculate the utilization of VMs. Utilization of a VM is used to calculate the corresponding reliability and energy consumption.

6.3.3 Execution of ReliableCloudSim

A UML sequence diagram representing communication between different entities of Reliable-Cloudsim during failure and recovery events is shown in Figure 6.5. Entity, FailureDatacenter triggers an event of a failure for an object of FailureHost representing physical host by calling a function *processFailureEvent()*. The controller then checks whether any VM is running on the host or not. In the case of VMs running on host, a flag representing state of failure for a VM is set

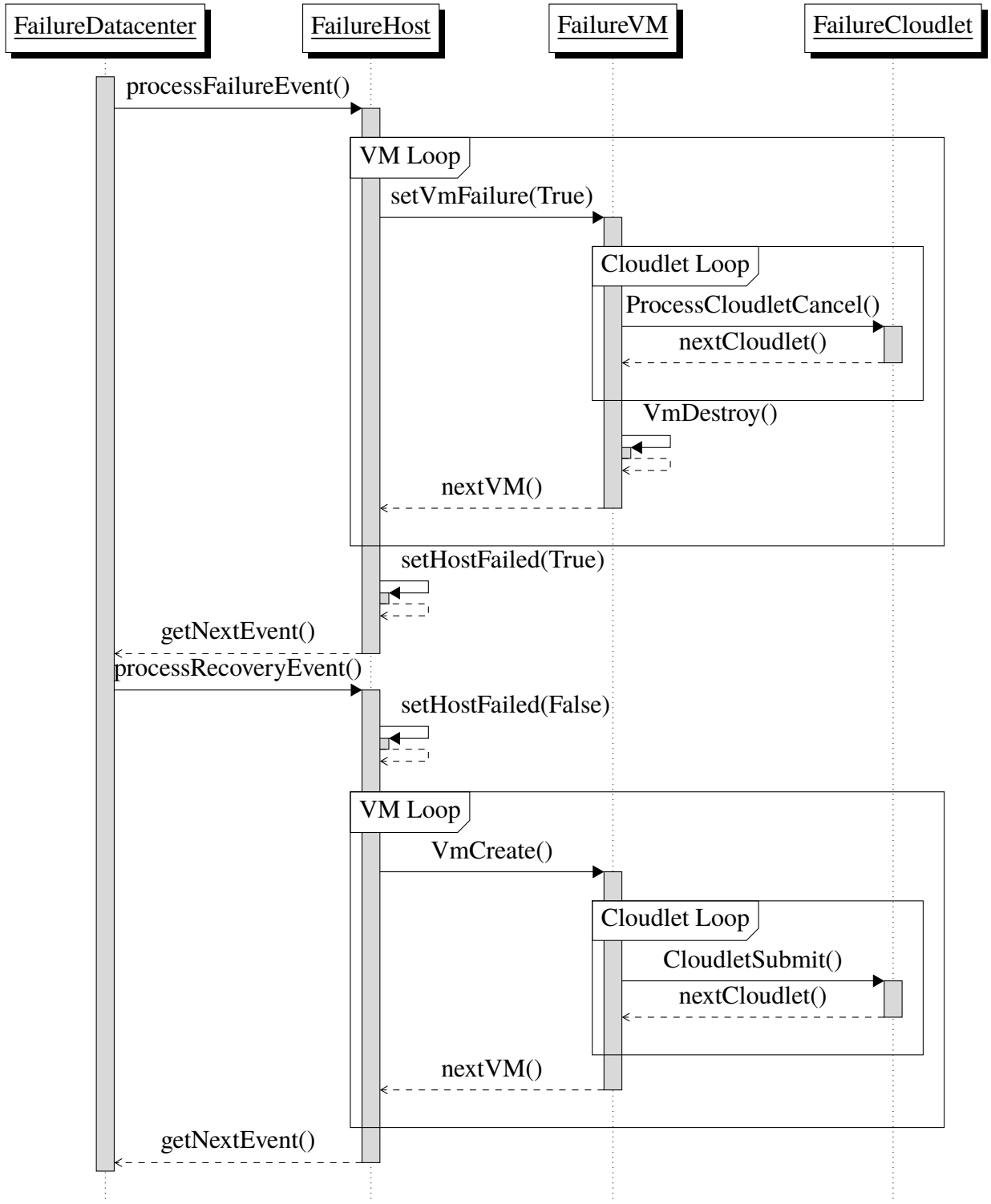


Figure 6.5: Sequence Diagram representing communication between ReliableCloudsim entities

to *True* by calling a function *setVmFailure()*. VM further calls *ProcessCloudletCancel()* function in order to cancel all the running cloudlets, if any. After cancelling all the running cloudlets, VM is destroyed by calling function *VmDestroy()*. The same procedure is repeated for all the VMs and corresponding cloudlets running on a failing host. After cancelling all the cloudlets and destroying VMs, the flag representing the failure state of a host is set to *True* by calling function *setHostFailed()*. In the last, the control will go back to an object of FailureDatacenter class in order to execute the next event.

In the case of recovering from a failure, FailureDatacenter entity triggers a call to function *processRecoveryEvent()* for a failed host. After receiving the request, the value of a flag representing failure state of a host changes from *True* to *False* by using a function *setHostFailed()*. Once the state is changed, the process of creation of failed VMs corresponding to recovering host starts by calling a function *VmCreate()*. This function further calls *CloudletSubmit()* function in order to create and submit the failed cloudlets corresponding to recovered VM. After creating and submitting all the cloudlets, next VM is created by using nextVM() function, if multiple VMs were running on failed host. After the creation of all the VMs and corresponding cloudlets and state of failed host is changed to available, control goes back to FailureDatacenter object and retrieves the next event needs to be executed by calling *getNextEvent()* function, if available.

6.4 Comparison of Simulated and Real Occurrence of Failures

In order to validate the accuracy of the simulation framework, the statistical comparison of the occurrence of failures on real infrastructure with the simulated occurrence of failures is done. For the comparison, failure traces gathered from Grid5000 and LANL computing infrastructure are obtained from Failure Trace Archive (FTA)[88]. FTA is an on-line public repository providing failure traces gathered from 26 different computing sites. Grid5000 failure dataset consists of the data gathered from 9 geographically distributed sites consisting of 15 different clusters, which is collected for 1.5 years between 2005-2006. Traces provided information about the failures and hardware configuration of approximately 1300 nodes installed across the sites. Whereas, traces

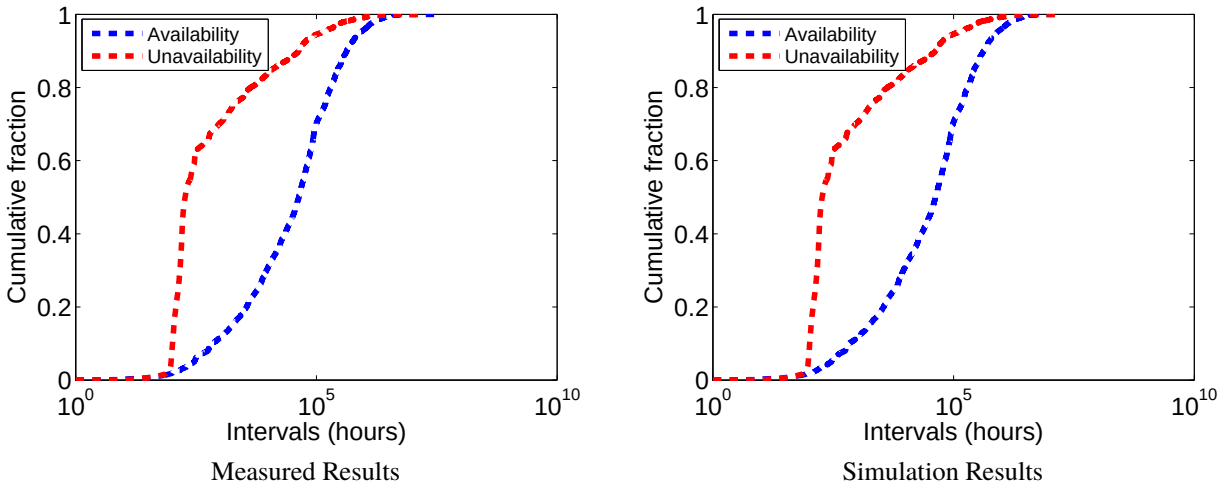


Figure 6.6: Measured Results vs Simulation Results for Grid5000 Failure Traces

from LANL systems were collected between year June 1996 to November 2005 and covers data from 23 high performance computing systems consisting of 4750 nodes in total.

The comparison of real and simulation results is performed in two phases. The first phase corresponds to simulation, where simulation environment is injected with the failures using failure traces and information about time to return (TTR) and time between failures (TBF) also know as unavailability and availability events, respectively is collected. Information about the physical resources (hosts) get provisioned during the simulation is also collected in terms of host IDs. The cumulative distribution functions (CDFs) of availability and unavailability events collected during the simulation is plotted. In the second phase, which corresponds to real scenario, the host information (host IDs) collected during the simulation is used to extract the available and unavailable events corresponding to each host from the traces. The information extraction from the traces is performed by using statistical tools developed using Matlab framework. After extracting the information, the CDFs are plotted for real scenario as well.

Figure 6.6 and 6.7 shows the results in the form of CDFs for both real execution termed as 'Measured' and ReliableCloudSim simulation termed as 'Simulation'. From the figures it can be seen clearly that CDF achieved for available and unavailable events from the traces gathered from

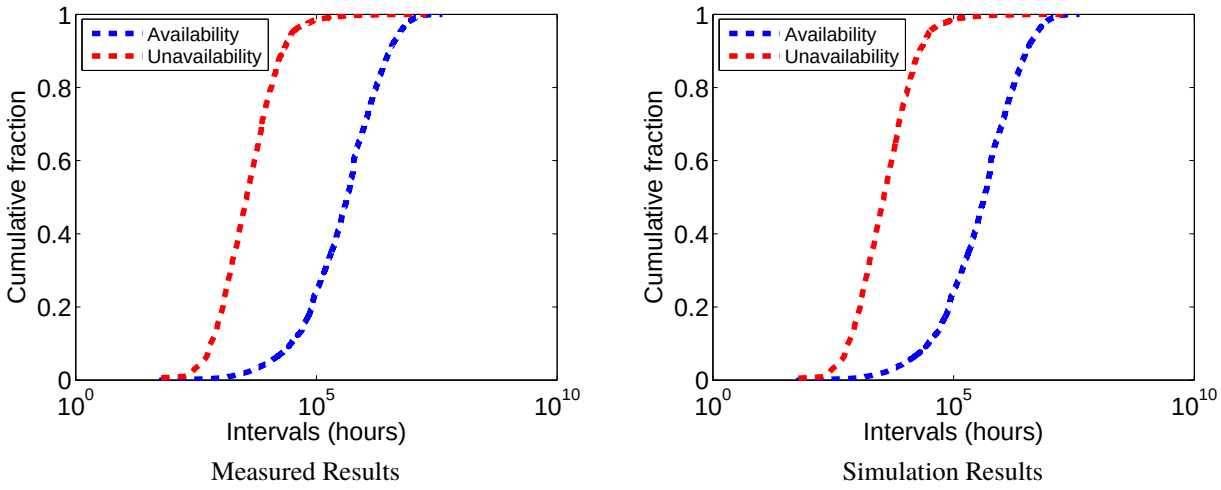


Figure 6.7: Measured Results vs Simulation Results for LANL Failure Traces

real infrastructure is closely followed by CDFs obtained from simulation results. The change in the CDFs for the both scenarios is very low, even negligible and similar behaviour is found for both of the used traces such as Grid5000 and LANL. This shows that ReliableCloudSim is effective to simulate the real life failure prone cloud computing system and can be used as a test bed framework by the researchers and cloud service providers to design and evaluate the resource management policies for a failure prone cloud computing environment.

6.5 Summary

In this chapter, a cloud computing deployment architecture to increase the reliability and energy efficiency is proposed. The proposed architecture considers the failure characteristics of the physical resources to take resource management decisions. The architecture is implemented as 'ReliableCloudSim', an extension of a well-known CloudSim simulator to simulate failure-prone cloud computing environment. The ReliableCloudSim provides a testbed to cloud service providers and researchers to design, test and improve resource management policies for a failure prone cloud computing environment before adopting them for real physical infrastructure. This will minimize

the chances of errors during real implementation. The use of our proposed framework will help the cloud service providers to draft better service level agreements (SLA) in order to minimize the losses and will help the service providers to expand their businesses in a prudent manner. The accuracy of the proposed framework is shown by plotting the CDFs of available and unavailable events obtained from the simulation and real infrastructure of Grid5000 and LANL computing sites. The plots of simulation CDFs are found to be similar to the CDF plots obtained from real infrastructure, which shows the accuracy of ReliableCloudSim to simulate the failure prone cloud computing environment.

Chapter 7

Conclusions and Future Directions

This chapter summarises the work done in this thesis to increase jointly the reliability and energy efficiency of cloud computing systems. The future directions and open research challenges emerged while carrying out this research are also discussed.

7.1 Conclusions and Discussion

Cloud computing systems help the users to reduce IT related operational costs and complexities by providing computing as a service. The service is provided by using easy to use portals hiding all the complex operations underneath, which users do not need to know about. While providing such an abstract view of the system, cloud computing systems have to perform many complex operations besides managing a large underlying infrastructure. Due to such complex operations, cloud computing systems are confronted by many challenges such as security, sustainability, reliability, resource management and energy efficiency. Among all described challenges, this thesis has laid out methods and mechanisms targeting a trade-off between reliability and energy efficiency of cloud computing systems. In order to achieve the objective of jointly increasing the reliability and energy-efficiency, various resource management policies incorporated with different fault-tolerance methods are proposed in this thesis. Further in this section, all the observations and conclusions of

the thesis are summarized.

It is identified that present literature works are either focused on the reliability techniques or energy efficiency methods in cloud computing and ignores the crucial trade-off existing between them. In response to bridging this gap, chapter 2 presents a thorough review of existing techniques for reliability and energy efficiency and their trade-off in cloud computing. Various types of failures are explored that drive researchers to design the mechanisms to make the cloud computing systems highly reliable. Chapter 2 has surveyed and critiqued a variety of methods aimed at increasing the reliability of cloud computing systems. The increase in the size and design complexity of clouds is resulting in huge energy consumption and enormous carbon footprints. This chapter also presented a comprehensive survey of all the energy management techniques used in cloud computing systems. The classifications are also discussed and provides in-depth taxonomies on resource failures, fault tolerance mechanisms and energy management mechanisms in cloud systems. It is observed that the adoption of mechanisms to provide reliability in cloud computing services has impacted the energy consumption of the system. Adding backup resources, running replicated systems, storing logs provide strong fault tolerance but also increase the energy consumption. While focusing on this trade-off between service reliability and energy consumption, the need for reliability-aware and energy-aware resource provisioning policies is identified to improve the availability of cloud services whilst simultaneously reducing its energy consumption.

Chapter 3 provides a mathematical model of both reliability and energy consumption in cloud computing systems and analyse their interplay. This chapter also proposes a formal method to calculate the finishing time of tasks running in a failure prone cloud computing environment using checkpointing as fault tolerance and without checkpointing. To achieve the objective of maximizing the reliability and minimizing the energy-consumption of cloud computing systems, three resource provisioning and virtual machine (VM) allocation policies, Reliability Aware Best Fit Decreasing (RABFD), Energy Aware Best Fit Decreasing (EABFD) and Reliability-Energy Aware Best Fit Decreasing (REABFD) using the proposed mathematical models are proposed. After performing an extensive simulation based study, it is concluded that if emphasis is given only to the energy optimization without considering reliability in a failure prone cloud computing environment, then

the results will be contrary to the expectations. Rather than reducing the energy consumption, the system ends up consuming more energy due to the energy losses incurred because of failure overheads. Among the proposed policies, Reliability-Energy Aware Best Fit Decreasing (REABFD) policy outperformed all the other policies and revealed that if both reliability and energy efficiency factors of resources are considered at the same time then both factors can be improved to a larger extent than being regulated individually.

VM consolidation is an important technique used in cloud computing systems to improve energy efficiency. Focusing solely on energy efficiency and ignoring failure characteristics of physical resources while performing VM consolidation in a failure prone cloud computing environment has adverse effects. If the failure characteristics of resources is ignored then running VMs may get consolidated to unreliable physical resources. This will cause more failures and recreations of VMs, thus increasing the energy consumption. To solve this problem, chapter 4 proposes a failure-aware VM consolidation mechanism, which takes the occurrence of failures and the hazard rate of physical resources into consideration before performing VM consolidation. A failure prediction technique based on exponential smoothing is proposed to trigger two fault tolerance mechanisms (VM migration and VM checkpointing). A simulation based evaluation of the proposed VM consolidation mechanism is conducted by using real failure traces. The results demonstrate that by using the combination of checkpointing and VM migration with the proposed failure-aware VM consolidation mechanism, the energy consumption of cloud computing system is reduced by 34% and reliability is improved by 12% while decreasing the occurrence of failures by 14%. From the results, it is concluded that while performing the VM consolidation in a failure prone cloud computing system, a significant improvement in terms of energy efficiency and reliability can be achieved by considering the failure characteristics of physical resources. In order to achieve higher fault tolerance in cloud computing systems, it is better to use the combination of reactive and proactive fault tolerance mechanisms rather than using them individually.

Dependence of computing resources on each other in cloud computing systems makes them prone to fail in a correlated manner which impacts service reliability and energy efficiency of the system. The occurrence of correlated failures in cloud computing systems lead to a service outage

and violation of Service Level Agreement (SLA), which cost huge to cloud service providers and users. Chapter 5 proposes many mechanisms for improving reliability and energy efficiency jointly under correlated failures in cloud computing systems. To well understand failure correlation, statistical cluster analysis techniques are applied to real failure traces. To predict the occurrence of failures, an average-based time series analysis, i.e., moving average method is used. Furthermore, mathematical models are built to represent reliability and energy consumption of cloud computing systems in the presence of correlated failures. These mathematical models are used to design fault-tolerant and energy-aware resource provisioning mechanisms/policies. To provide fault tolerance and avoidance, both reactive (checkpointing) and proactive (VM migration) mechanisms are used, which get triggered on the basis of failure prediction results. In order to reduce the energy consumption dynamically, a correlated failure-aware VM consolidation policy is also proposed. A simulation based study of the proposed resource management policies and fault tolerance mechanisms is conducted by using real failure traces. Verified by the extensive simulation study, it is concluded that by considering correlated occurrence of failures during resource provisioning and VM allocation in cloud computing systems, the service downtime or interruption is reduced significantly by 34% and turnaround time is reduced by 11% in comparison to the environments with the assumption of independent occurrence of failures. This reduction in return has improved QoS by reducing the SLA violations and increased the energy efficiency by approximately 20%. The interplay between the reliability and energy consumption of cloud computing systems is improved by 14% by considering failure correlation in association with the proposed failure avoidance and fault tolerance mechanisms.

Finally, in chapter 6 a cloud computing deployment architecture to increase the reliability and energy efficiency of cloud computing systems is proposed. The architecture considers the failure characteristics of the physical resources to take any resource management decisions. The architecture is implemented by extending CloudSim simulator to simulate failure-prone cloud computing environment, i.e., 'ReliableCloudSim'. This architecture provides a test bed to cloud service providers and cloud reserchers using which new resource management policies can be designed, tested and improved in a failure prone cloud computing environment before implementing

them on real infrastructure. This will minimize the chances of the occurrence of application errors during live implementation, which can result in service failures. Use of proposed framework will help the cloud service providers to draft better service level agreements (SLA) in order to minimize the losses and improve the reputation of cloud services and will help the service providers to expand their businesses with less operational expenses. The accurracy of the proposed framework is shown by ploting the CDFs of available and unavailable events obtained from the simulation and real infrastructure of Grid5000 and LANL computing sites. The plots of simulation CDF is found to be similar to CDF of real infrastructure which shows the effectivenes of 'ReliableCloudSim' to simulate the failure prone cloud computing environment.

7.2 Future Directions

In this thesis, the problem of reliable and energy efficient resource provisioning in cloud computing systems is addressed. However, there are research challenges that are remaining opened and can be explored in future.

7.2.1 Efficient Reliability Management of Cloud Storage

Cloud storage systems are used intensively for storing data generated by Big data applications. However, cloud storage systems are also reported as the most failure prone devices among all the components of a cloud computing infrastructure. Improving fault tolerance in cloud storage systems for Big data applications is a significant challenge. Replication is a simple solution to provide fault tolerance and improve data reliability. However, for Big Data applications replication involves a large amount of storage overhead which makes the solution very inefficient and expensive. Recently cloud data centres started adapting erasure coding to improve the reliability of Big Data applications with less storage overhead. The major issue with erasure coding is high resource consumption in terms of disk I/O, network bandwidth/traffic and energy upon failure. Such overheads provide a good scope of improvement and motivation to researchers to improve the data reliability with less

storage overhead and operational cost. Hybrid techniques combining data replication and erasure coding supported by accurate failure prediction needs to developed, which will significantly reduce the network traffic and improve the performance of Big data application with less storage overhead.

7.2.2 Reliability-Aware Resource Management for Fog Computing

With the recent hardware advancements, mobile devices such as phones and tablets, have achieved better memory management and processing power. Besides the basic hardware units these devices are also equipped with different sensors, cameras and other data transmission and receiving units. Such advancements has led to the usage of these devices in variety of applications such as mobile commerce, social networking and many others [25], which tend to generate data intensively and requires extra bandwidth and processing capability in order to get their requests processed, timely. Such resource requirements gave birth to a new domain of cloud computing, which is Fog computing. Fog computing paradigm provides an extra layer of edge devices such as networking equipments with processing capabilities between the users and cloud computing infrastructure. These edge devices are placed near to the data generation sources and performs general purpose computing in order to reduce the load on cloud computing infrastructure and reduce the latency for real time and streaming applications, such as fitness tracking and traffic monitoring. Due to the load decomposition between the edge devices and cloud computing infrastructure, the edge devices are assigned with extra responsibilities which were not a part of traditional cloud computing paradigm. Such extra responsibilities increases the workload on the edge devices and increases their proneness towards failures. For example, the deployment topologies used in cloud computing paradigm will be more prone to single point failures in fog computing paradigm. Such challenges can be overcome by designing new topologies and by employing complex and smart resource management policies. This opens new research questions to design reliability aware resource management policies for Fog computing while taking various factors such as application structure, context information, topology designs, mobile-device energy consumption and network status in order to provide an uninterruptible service to users.

7.2.3 Reliable Software Defined Networks (SDN)

As a new technology, importing of SDN paradigm to cloud computing systems can bring along new reliability challenges that researchers need to explore. The occurrence of failures can be different from the traditional networks in ways that have not been experienced or expected. For instance, a failure at SDN controller can jeopardize the working of a system and can even bring the whole cloud computing system down. In order to avoid such situations, more research is required by gathering the information about data spikes and source of data, which will be fed to employed machine learning techniques in order to model the occurrence of failures and measures will be taken to prevent their occurrence to ensure high reliability of the system.

7.2.4 Failure Prediction Accuracy

Inaccurate failure prediction triggers the fault tolerance measures such as VM migration, VM checkpointing and VM replication, unnecessarily and imposes extra overheads. This reduces the reliability and energy efficiency of the system and increases SLA violations by increasing the execution duration of an application. Such unnecessary triggering of fault tolerance measures can be avoided by improving the accuracy of failure prediction algorithms by employing deep and machine learning techniques producing results by using failure datasets collected historically. Accurate failure prediction will also help to realize the future technologies such as Internet of Things (IoT), Edge and Fog computing in an efficient way by using fault tolerance efficiently and by reducing extra overheads generated because of wrong failure prediction, which will reduce the extra processing required by computationally limited edge computing devices.

7.2.5 Reliable Distributed Cloud Computing Systems

The introduction of container technology, SDNs, Network Functions Virtualization (NVF) and other portability configuration tools reduced the complexity of realization of distributed cloud computing systems. However, there are number of research challenges still remains opened.

Reliability aware resource provisioning and VM allocation in distributed cloud environment in one of such challenges. Meta-schedulers targeting energy efficiency, carbon footprint, cost and response time are proposed in the literature. However, failure-aware meta-scheduler taking failure characteristics at data center level into account while making decisions about resource provisioning and VM allocation is not explored and proposed yet. Such meta-schedulers have great potential to make the distributed cloud computing systems immune from the occurrence of failures and provides a great motivation to design novel distributed cloud resource management solutions in order to provide an uninterruptible services to users in a reliable manner.

Bibliography

[1] Quality Excellence for Suppliers of Telecommunications Forum (QuEST Forum), 2010.

[2] Sherly Abraham and InduShobha Chengalur-Smith. "An overview of social engineering malware: Trends, tactics, and implications". In: *Technology in Society* 32.3 (2010), pp. 183–196.

[3] Air and Lloyd. *Failure of a top cloud service provider could cost US economy 15 billion*. 2018. URL: http://www.air-worldwide.com/Press-Releases/Failure-of-a-top-cloud-service-provider-could-cost-US-economy-15-billion/ (visited on 08/05/2018).

[4] Sherif Akoush et al. "Predicting the performance of virtual machine migration". In: *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2010 IEEE International Symposium on*. IEEE. 2010, pp. 37–46.

[5] Mohammed AlZain et al. "Cloud computing security: from single to multi-clouds". In: *45th Hawaii International Conference on System Science (HICSS)*. IEEE. Maui, HI, USA, 2012, pp. 5490–5499.

[6] Amazon. *AWS and Sustainability*. URL: https://aws.amazon.com/about-aws/sustainability/ (visited on 08/05/2015).

[7] Amazon. *Summary of the Amazon S3 Service Disruption in the Northern Virginia (US-EAST-1) Region*. 2017. URL: https://aws.amazon.com/message/41926 (visited on 04/30/2018).

[8] David G Andersen et al. “FAWN: A fast array of wimpy nodes”. In: *Proceedings of the 22nd ACM Symposium on Operating Systems Principles*. ACM. Big Sky, MT, USA, 2009, pp. 1–14.

[9] Sebastian Anthony. “Microsoft now has one million servers–less than Google, but more than Amazon, says Ballmer”. In: *ExtremeTech. ExtremeTech* 19 (2013).

[10] Esmaeil Atashpaz-Gargari and Caro Lucas. “Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition”. In: *Congress on Evolutionary Computation (CEC)*. IEEE. Singapore, 2007, pp. 4661–4667.

[11] Anju Bala and Inderveer Chana. “Autonomic fault tolerant scheduling approach for scientific workflows in Cloud computing”. In: *Concurrent Engineering* 23.1 (2015), pp. 27–39.

[12] Paul Barham et al. “Xen and the art of virtualization”. In: *ACM SIGOPS operating systems review*. Vol. 37. 5. ACM. 2003, pp. 164–177.

[13] Luiz André Barroso, Jimmy Clidaras, and Urs Hölzle. “The datacenter as a computer: An introduction to the design of warehouse-scale machines”. In: *Synthesis lectures on computer architecture* 8.3 (2013), pp. 1–154.

[14] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. “Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing”. In: *Future generation computer systems* 28.5 (2012), pp. 755–768.

[15] Anton Beloglazov et al. “A taxonomy and survey of energy-efficient data centers and cloud computing systems”. In: *Advances in computers* 82.2 (2011), pp. 47–111.

[16] SPEC Benchmarks. *Standard performance evaluation corporation*. 2000.

[17] Nicolas Bonvin, Thanasis G Papaioannou, and Karl Aberer. “A self-organized, fault-tolerant and scalable replication scheme for cloud storage”. In: *Proceedings of the 1st ACM symposium on Cloud computing*. ACM. Indianapolis, IN, USA, 2010, pp. 205–216.

[18] Tom Bostoen, Sape Mullender, and Yolande Berbers. “Power-reduction techniques for data-center storage systems”. In: *ACM Computing Surveys (CSUR)* 45.3 (2013), p. 33.

[19] David J Bradley, Richard E Harper, and Steven W Hunter. “Workload-based power management for parallel computer systems”. In: *IBM Journal of Research and Development* 47.5.6 (2003), pp. 703–718.

[20] Tracy D Braun et al. “A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems”. In: *Journal of Parallel and Distributed computing* 61.6 (2001), pp. 810–837.

[21] Gregory Bresiger. *Knight Capital computer meltdown just waiting to happen*. 2013. URL: http://nypost.com/2013/10/26/knight-capital-computer-meltdown-just-waiting-to-happen/ (visited on 03/08/2015).

[22] Marc J Brooker et al. *Resource allocation to reduce correlated failures*. US Patent App. 15/665,019. 2017.

[23] Uri Budnik. *Lessons Learned from Recent Cloud Outages*. 2013. URL: https://www.rightscale.com/blog/enterprise-cloud-strategies/lessons-learned-recent-cloud-outages (visited on 08/05/2015).

[24] Jennifer Burge, Parthasarathy Ranganathan, and Janet L Wiener. “Cost-aware scheduling for heterogeneous enterprise machines (CASHEM)”. In: *International Conference on Cluster Computing*. IEEE. Austin, TX, USA, 2007, pp. 481–487.

[25] Rajkumar Buyya et al. “A manifesto for future generation cloud computing: Research directions for the next decade”. In: *ACM Computing Surveys (CSUR)* 51.5 (2018), p. 105.

[26] Rodrigo N Calheiros and Rajkumar Buyya. “Energy-efficient scheduling of urgent bag-of-tasks applications in clouds through DVFS”. In: *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*. IEEE. 2014, pp. 342–349.

[27] Rodrigo N Calheiros et al. “CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms”. In: *Software: Practice and experience* 41.1 (2011), pp. 23–50.

[28] Franck Cappello et al. “Grid5000: a nation wide experimental grid testbed”. In: *International Journal on High Performance Computing Applications* 20.4 (2006), pp. 481–494.

[29] Adrian M Caulfield, Laura M Grupp, and Steven Swanson. “Gordon: using flash memory to build fast, power-efficient clusters for data-intensive applications”. In: *ACM Sigplan Notices* 44.3 (2009), pp. 217–228.

[30] Yiyu Chen et al. “Managing server energy and operational costs in hosting centers”. In: *ACM SIGMETRICS Performance Evaluation Review* 33.1 (2005), pp. 303–314.

[31] Christopher Clark et al. “Live Migration of Virtual Machines”. In: *Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2*. NSDI’05. Berkeley, CA, USA: USENIX Association, 2005, pp. 273–286.

[32] Garry Cook and Jodle Van Horn. “How dirty is your Data? A look at the Energy Choices that Power Cloud Computing”. In: (2011), pp. 1–36.

[33] International Data Corporation. *Worldwide Public Cloud Services Spending Forecast to Reach 160 Billion This Year, According to IDC*. 2018. URL: `https://www.idc.com/getdoc.jsp?containerId=prUS43511618` (visited on 08/04/2018).

[34] Oracle Corporation. *Cloud: opening up the road to Industry 4.0*. 2018. URL: `https://www.oracle.com/uk/cloud/paas/features/next-industrial-revolution.html` (visited on 08/04/2018).

[35] Brendan Cully et al. “Remus: High availability via asynchronous virtual machine replication”. In: *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*. San Francisco. 2008, pp. 161–174.

[36] Yuan-Shun Dai et al. “Cloud service reliability: Modeling and analysis”. In: *15th IEEE Pacific Rim International Symposium on Dependable Computing*. Citeseer. 2009, pp. 1–17.

[37] John T Daly. “A higher order estimate of the optimum checkpoint interval for restart dumps”. In: *Future Generation Computer Systems* 22.3 (2006), pp. 303–312.

[38] Howard David et al. “Memory power management via dynamic voltage/frequency scaling”. In: *Proceedings of the 8th ACM international conference on Autonomic computing*. ACM. Karlsruhe, Germany, 2011, pp. 31–40.

[39] Pierre Delforge. *America’s Data Centers Consuming and Wasting Growing Amounts of Energy*. 2015. URL: http://www.vox.com/2014/12/14/7387945/sony-hack-explained (visited on 03/08/2015).

[40] Wei Deng et al. “Lifetime or energy: Consolidating servers with reliability control in virtualized cloud datacenters”. In: *4th International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE. Taipei, Taiwan, 2012, pp. 18–25.

[41] Srinivas Devadas and Sharad Malik. “A survey of optimization techniques targeting low power VLSI circuits”. In: *Proceedings of the 32nd annual ACM/IEEE Design Automation Conference*. ACM/IEEE. San Francisco, CA, USA, 1995, pp. 242–247.

[42] Sheng Di et al. “LogAider: A tool for mining potential correlations of HPC log events”. In: *Cluster, Cloud and Grid Computing (CCGRID), 2017 17th IEEE/ACM International Symposium on*. IEEE. 2017, pp. 442–451.

[43] Availability Digest. *IT as a service: From build to consume*. 2013. URL: http://www.availabilitydigest.com/public-articles/0811/azure.pdf (visited on 08/05/2018).

[44] Andrzej Duda. “The effects of checkpointing on program execution time”. In: *Information Processing Letters* 16.5 (1983), pp. 221–229.

[45] Chad Eschinger Joanne M. Correia Laurie F. Wurster Tom Eid Vennecia K Liu Chis Pang Dan Sommer Hai Hong Swinehart Jie Zhang Morgan Yeates Warren Bell Gregor Petri Ruggero Contu Ed Anderson Yanna Dharmasthira. *Forecast Overview: Public Cloud*

Services, Worldwide, 2Q13 Update. 2013. URL: https://www.gartner.com/doc/2581118/forecast-overview-public-cloud-services (visited on 12/11/2014).

[46] I.P. Egwutuoha et al. "Energy Efficient Fault Tolerance for High Performance Computing (HPC) in the Cloud". In: *Sixth International Conference on Cloud Computing (CLOUD)*. IEEE. Santa Clara, CA, USA, 2013, pp. 762–769. DOI: 10.1109/CLOUD.2013.69.

[47] Nosayba El-Sayed and Bianca Schroeder. "To checkpoint or not to checkpoint: Understanding energy-performance-i/o tradeoffs in hpc checkpointing". In: *2014 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE. 2014, pp. 93–102.

[48] Elmootazbellah Nabil Elnozahy et al. "A survey of rollback-recovery protocols in message-passing systems". In: *ACM Computing Surveys (CSUR)* 34.3 (2002), pp. 375–408.

[49] Arul Elumalai, Sid Tandon, and Irina Starikova. *IT as a service: From build to consume*. 2016. URL: https://www.mckinsey.com/industries/high-tech/our-insights/IT-as-a-service-From-build-to-consume (visited on 08/05/2018).

[50] Christian Engelmann and Al Geist. "Super-scalable algorithms for computing on 100,000 processors". In: *International Conference on Computational Science*. Springer. 2005, pp. 313–321.

[51] Hamid Reza Faragardi et al. "Towards energy-aware resource scheduling to maximize reliability in cloud computing systems". In: *10th International Conference on High Performance Computing and Communications (HPCC)*. IEEE. Dalian, China, 2013, pp. 1469–1479.

[52] Daniel Ford et al. "Availability in Globally Distributed Storage Systems." In: *Osdi*. Vol. 10. 2010, pp. 1–7.

[53] Thomas Fox-Brewster. *Why You Shouldn't Panic About Dropbox Leaking 68 Million Passwords*. 2016. URL: https://www.forbes.com/sites/thomasbrewster/2016/08/31/dropbox-hacked-but-its-not-that-bad/ (visited on 08/11/2015).

[54] Song Fu. “Failure-aware resource management for high-availability computing clusters with distributed virtual machines”. In: *Journal of Parallel and Distributed Computing* 70.4 (2010), pp. 384–393.

[55] Song Fu and Cheng-Zhong Xu. “Exploring event correlation for failure prediction in coalitions of clusters”. In: *Proceedings of the Conference on Supercomputing (SC’07)*. ACM/IEEE. Reno, NV, USA, 2007, pp. 1–12.

[56] Matthieu Gallet et al. “A model for space-correlated failures in large-scale distributed systems”. In: *Euro-Par 2010-Parallel Processing*. Ischia, Italy: Springer, 2010, pp. 88–100.

[57] Anshul Gandhi et al. “Minimizing data center sla violations and power consumption via hybrid resource provisioning”. In: *International Green Computing Conference and Workshops (IGCC)*. IEEE. Orlando, FL, USA, 2011, pp. 1–8.

[58] Anshul Gandhi et al. “Optimal power allocation in server farms”. In: *ACM SIGMETRICS Performance Evaluation Review* 37.1 (2009), pp. 157–168.

[59] Aiqiang Gao and Luhong Diao. “Lazy update propagation for data replication in cloud computing”. In: *5th International Conference on Pervasive Computing and Applications (ICPCA)*. IEEE. maribor, slovenia, 2010, pp. 250–254.

[60] Saurabh Kumar Garg, Chee Shin Yeo, and Rajkumar Buyya. “Green cloud framework for improving carbon efficiency of clouds”. In: *Euro-Par 2011 Parallel Processing* 6852 (2011), pp. 491–502.

[61] Saurabh Kumar Garg et al. “Environment-conscious scheduling of HPC applications on distributed cloud-oriented data centers”. In: *Journal of Parallel and Distributed Computing* 71.6 (2011), pp. 732–749.

[62] Peter Garraghan et al. “An Analysis of Failure-Related Energy Waste in a Large-Scale Cloud Environment”. In: *IEEE Transactions on Emerging Topics in Computing* 2.2 (2014), pp. 166–180.

[63] Sarah Gelper, Roland Fried, and Christophe Croux. “Robust forecasting with exponential and Holt–Winters smoothing”. In: *Journal of forecasting* 29.3 (2010), pp. 285–300.

[64] Mahboobeh Ghorbani et al. “Prediction and control of bursty cloud workloads: a fractal framework”. In: *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis*. ACM. New Delhi, India, 2014, pp. 1–9.

[65] Maurizio Giacobbe et al. “Towards energy management in cloud federation: a survey in the perspective of future sustainable and cost-saving strategies”. In: *Computer Networks* 91 (2015), pp. 438–452.

[66] Google. *Google Compute Engine Incident #15056*. 2015. URL: https://status.cloud.google.com/incident/compute/15056 (visited on 08/26/2018).

[67] Rachid Guerraoui and André Schiper. “Fault-tolerance by replication in distributed systems”. In: *Reliable Software Technologies Ada-Europe’96*. Springer. Montreux, Switzerland, 1996, pp. 38–57.

[68] Sudhanva Gurumurthi et al. “Reducing disk power consumption in servers with DRPM”. In: *Computer* 12 (2003), pp. 59–66.

[69] Alex Gutman. *Downtime, Outages and Failures-Understanding their True Costs*. 2012. URL: http://www.evolven.com/blog/downtime-outages-and-failures-understanding-their-true-costs.html (visited on 03/08/2015).

[70] Abdul Hameed et al. “A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems”. In: *Computing* (2014), pp. 1–24.

[71] Michael R Hines, Umesh Deshpande, and Kartik Gopalan. “Post-copy live migration of virtual machines”. In: *ACM SIGOPS operating systems review* 43.3 (2009), pp. 14–26.

[72] IDG. *Understand Organizations Cloud Computing Plans*. 2018. URL: https://resources.idg.com/download/executive-summary/cloud-computing-2018 (visited on 08/21/2018).

[73] Ponemon Institute. *Cost of Data Center Outages*. 2016. URL: https://planetaklimata.com.ua/instr/Liebert_Hiross/Cost_of_Data_Center_Outages_2016_Eng.pdf.

[74] Ponemon Institute. *The Value of Artificial Intelligence in Cybersecurity*. 2018. URL: https://www.themsphub.com/app/uploads/2018/09/ibm-ai-report-final-1_41017541USEN.pdf.

[75] Alexandru Iosup et al. "The performance of bags-of-tasks in large-scale distributed systems". In: *Proceedings of the 17th international symposium on High performance distributed computing*. ACM. 2008, pp. 97–108.

[76] Sadeka Islam et al. "Empirical prediction models for adaptive resource provisioning in the cloud". In: *Future Generation Computer Systems* 28.1 (2012), pp. 155–162.

[77] Ravishankar K Iyer and David J Rossetti. "A measurement-based model for workload dependence of CPU errors". In: *IEEE Transactions on Computers* 100.6 (1986), pp. 511–519.

[78] Bahman Javadi, Jemal Abawajy, and Rajkumar Buyya. "Failure-aware resource provisioning for hybrid Cloud infrastructure". In: *Journal of parallel and distributed computing* 72.10 (2012), pp. 1318–1331.

[79] Bahman Javadi, Parimala Thulasiraman, and Rajkumar Buyya. "Enhancing performance of failure-prone clusters by adaptive provisioning of cloud resources". In: *The Journal of Supercomputing* 63.2 (2013), pp. 467–489.

[80] Bahman Javadi et al. "Discovering statistical models of availability in large distributed systems: An empirical study of seti@ home". In: *IEEE Transactions on Parallel and Distributed Systems* 22.11 (2011), pp. 1896–1903.

[81] Ravi Jhawar, Vincenzo Piuri, and Marco Santambrogio. "Fault tolerance management in cloud computing: A system-level perspective". In: *Systems Journal, IEEE* 7.2 (2013), pp. 288–297.

[82] Penny Jones. *Overheating brings down Microsoft data center*. 2013. URL: http://www.datacenterdynamics.com/focus/archive/2013/03/overheating-brings-down-microsoft-data-center (visited on 08/05/2015).

[83] Daeyong Jung et al. "VM migration for fault tolerance in spot instance based cloud computing". In: *Grid and Pervasive Computing* 7861 (2013), pp. 142–151.

[84] Tarandeep Kaur and Inderveer Chana. "Energy efficiency techniques in cloud computing: A survey and taxonomy". In: *ACM Computing Surveys (CSUR)* 48.2 (2015), p. 22.

[85] Sean Keach. *Twitter DOWN social network back online after today's mystery outage*. 2018. URL: https://www.thesun.co.uk/tech/6074848/twitter-down-offline-not-working (visited on 04/18/2018).

[86] Atefeh Khosravi, Saurabh Kumar Garg, and Rajkumar Buyya. "Energy and carbon-efficient placement of virtual machines in distributed cloud data centers". In: *Euro-Par 2013 Parallel Processing* 8097 (2013), pp. 317–328.

[87] Jinoh Kim and Doron Rotem. "FREP: Energy proportionality for disk storage using replication". In: *Journal of Parallel and Distributed Computing* 72.8 (2012), pp. 960–974.

[88] Derrick Kondo et al. "The failure trace archive: Enabling comparative analysis of failures in diverse distributed systems". In: *10th International Conference on Cluster, Cloud and Grid Computing (CCGrid)*. IEEE/ACM. Melbourne, Victoria, Australia, 2010, pp. 398–407.

[89] Masaaki Kondo and Hiroshi Nakamura. "Dynamic processor throttling for power efficient computations". In: *Lecture notes in computer science* 3471 (2005), pp. 120–134.

[90] Negin Kord and Hamed Haghighi. "An energy-efficient approach for virtual machine placement in cloud based data centers". In: *5th Conference on Information and Knowledge Technology (IKT)*. IEEE. Shiraz, Iran, 2013, pp. 44–49.

[91] Pierre L'Ecuyer and Jacques Malenfant. "Computing optimal checkpointing strategies for rollback and recovery systems". In: *IEEE Transactions on Computers* 37.4 (1988), pp. 491–496.

[92] Etienne Le Sueur and Gernot Heiser. “Dynamic voltage and frequency scaling: The laws of diminishing returns”. In: *Proceedings of the international conference on Power aware computing and systems*. USENIX Association. Vancouver, BC, Canada, 2010, pp. 1–5.

[93] Trung Le and David Wright. “Scheduling workloads in a network of datacentres to reduce electricity cost and carbon footprint”. In: *Sustainable Computing: Informatics and Systems* 5 (2015), pp. 31–40.

[94] Young Choon Lee and Albert Y Zomaya. “Energy efficient utilization of resources in cloud computing systems”. In: *The Journal of Supercomputing* 60.2 (2012), pp. 268–280.

[95] Laurent Lefèvre and Anne-Cécile Orgerie. “Designing and evaluating an energy efficient cloud”. In: *The Journal of Supercomputing* 51.3 (2010), pp. 352–373.

[96] Pierre Lemarinier et al. “Improved message logging versus improved coordinated checkpointing for fault tolerant MPI”. In: *International Conference on Cluster Computing*. IEEE. San Diego, CA, USA, 2004, pp. 115–124.

[97] Xiang Li et al. “Holistic energy and failure aware workload scheduling in Cloud datacenters”. In: *Future Generation Computer Systems* 78 (2018), pp. 887–900.

[98] Min Yeol Lim et al. “Padd: Power aware domain distribution”. In: *29th International Conference on Distributed Computing Systems (ICDCS’09)*. IEEE. Montreal, Quebec, Canada, 2009, pp. 239–247.

[99] JoÃčo Marques Lima. *Data Centres Of The World Will Consume 1/5 Of Earth’s Power By 2025*. 2017. URL: `https://resources.idg.com/download/executive-summary/cloud-computing-2018` (visited on 08/15/2018).

[100] Jia-Chun Lin, Fang-Yie Leu, and Ying-ping Chen. “Analyzing job completion reliability and job energy consumption for a general MapReduce infrastructure”. In: *Journal of High Speed Networks* 19.3 (2013), pp. 203–214.

[101] Haikun Liu et al. “Live migration of virtual machine based on full system trace and replay”. In: *Proceedings of the 18th ACM international symposium on High performance distributed computing*. ACM. Munich, Germany, 2009, pp. 101–110.

[102] Fei Ma, Feng Liu, and Zhen Liu. “Live virtual machine migration based on improved pre-copy approach”. In: *IEEE International Conference on Software Engineering and Service Sciences*. IEEE. 2010, pp. 230–233.

[103] Violeta Medina and Juan Manuel García. “A survey of migration mechanisms of virtual machines”. In: *ACM Computing Surveys (CSUR)* 46.3 (2014), p. 30.

[104] Mohammed el Mehdi Diouri et al. “Energy considerations in checkpointing and fault tolerance protocols”. In: *IFIP International Conference on Dependable Systems and Networks Workshops (DSN)*. IEEE. Boston, MA, USA, 2012, pp. 1–6.

[105] David Meisner, Brian T Gold, and Thomas F Wenisch. “PowerNap: eliminating server idle power”. In: *ACM sigplan notices*. Vol. 44. 3. ACM. 2009, pp. 205–216.

[106] Esteban Meneses, Osman Sarood, and Laxmikant V Kalé. “Assessing energy efficiency of fault tolerance protocols for HPC systems”. In: *Computer Architecture and High Performance Computing (SBAC-PAD), 2012 IEEE 24th International Symposium on*. IEEE. 2012, pp. 35–42.

[107] Bakhta Meroufel and Ghalem Belalem. “Adaptive time-based coordinated checkpointing for cloud computing workflows”. In: *Scalable Computing: Practice and Experience* 15.2 (2014), pp. 153–168.

[108] Hugo Meyer, Dolores Rexachs, and Emilio Luque. “Hybrid Message Logging. Combining advantages of Sender-based and Receiver-based approaches”. In: *Procedia Computer Science* 29 (2014), pp. 2380–2390.

[109] Mohand Mezmaz et al. “A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems”. In: *Journal of Parallel and Distributed Computing* 71.11 (2011), pp. 1497–1508.

[110] James W Mickens and Brian D Noble. “Exploiting availability prediction in distributed systems”. In: *(NSDI’06), 3rd Symposium on Networked Systems Design and Implementation*. San Jose, CA, USA, 2006, pp. 73–86.

[111] Adams Mike et al. “An introduction to designing reliable cloud services”. In: *Trustworthy Computing* (2014), pp. 2–14.

[112] Rich Miller. *218000 Servers in Microsoft Data Centers*. 2008. URL: https://www.datacenterknowledge.com/archives/2008/08/14/218000-servers-in-microsoft-data-centers (visited on 08/23/2015).

[113] Dejan S Milojičić et al. “Process migration”. In: *ACM Computing Surveys (CSUR)* 32.3 (2000), pp. 241–299.

[114] Douglas C Montgomery, Cheryl L Jennings, and Murat Kulahci. *Introduction to time series analysis and forecasting*. John Wiley & Sons, 2015.

[115] Tung Nguyen and Weisong Shi. “Improving resource efficiency in data centers using reputation-based resource selection”. In: *Green Computing Conference, International*. Chicago, IL, USA, Aug. 2010, pp. 389–396. DOI: 10.1109/GREENCOMP.2010.5598290.

[116] Adam J Oliner et al. “Probabilistic qos guarantees for supercomputing systems”. In: *Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on*. IEEE. 2005, pp. 634–643.

[117] Sarah Perez. *Microsoft and Yahoo Confirm Search Outages*. 2015. URL: http://techcrunch.com/2015/01/02/following-bing-coms-brief-outage-search-yahoo-com-goes-down-too/ (visited on 03/08/2015).

[118] Jorge E Pezoa and Majeed M Hayat. “Reliability of heterogeneous distributed computing systems in the presence of correlated failures”. In: *IEEE Transactions on Parallel and Distributed Systems* 25.4 (2014), pp. 1034–1043.

[119] Ian Philp. “Software failures and the road to a petaflop machine”. In: *HPCRI: 1st Workshop on High Performance Computing Reliability Issues, in Proceedings of the 11th International Symposium on High Performance Computer Architecture (HPCA-11)*. San Francisco, California, USA, 2005, pp. 125–128.

[120] Eduardo Pinheiro, Ricardo Bianchini, and Cezary Dubnicki. “Exploiting redundancy to conserve energy in storage systems”. In: *ACM SIGMETRICS Performance Evaluation Review* 34.1 (2006), pp. 15–26.

[121] James S Plank and Wael R Elwasif. “Experimental assessment of workstation failures and their impact on checkpointing systems”. In: *Twenty-Eighth Annual International Symposium on Fault-Tolerant Computing*. IEEE. Munich, Germany, 1998, pp. 48–57.

[122] Brain Proffitt. *Software-as-a-Service The Dirty Little Secrets Of SaaS*. 2013. URL: `https://readwrite.com/2013/03/05/software-as-a-service-the-dirty-little-secrets-of-saas/` (visited on 08/10/2015).

[123] Xiwei Qiu et al. “A hierarchical correlation model for evaluating reliability, performance, and power consumption of a cloud service”. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 46.3 (2016), pp. 401–412.

[124] RT. *Facebook and Instagram outages reported across US and Europe*. 2018. URL: `https://on.rt.com/8xrg` (visited on 01/26/2018).

[125] RT. *This is not a drill! Netflix outages trigger ’doomsday’ distress after Facebook and Instagram issues*. 2018. URL: `https://on.rt.com/8xr` (visited on 01/26/2018).

[126] RT. *Whatsapp down in many parts of the world amid New Year celebrations*. 2018. URL: `https://on.rt.com/8vzq` (visited on 01/01/2018).

[127] Sampath Rangarajan, Sachin Garg, and Yennun Huang. “Checkpoints-on-demand with active replication”. In: *Seventeenth Symposium on Reliable Distributed Systems*. IEEE. West Lafayette, Indiana, USA, 1998, pp. 75–83.

[128] RightScale. *State of the Cloud Report*. 2017. URL: https://assets.rightscale.com/uploads/pdfs/RightScale-2017-State-of-the-Cloud-Report.pdf (visited on 03/12/2018).

[129] Élisson Rocha et al. "Analyzing the impact of power infrastructure failures on cloud application availability". In: *Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference on*. IEEE. 2017, pp. 1746–1751.

[130] Felix Salfner, Maren Lenk, and Miroslaw Malek. "A survey of online failure prediction methods". In: *ACM Computing Surveys (CSUR)* 42.3 (2010), p. 10.

[131] Altino M Sampaio and Jorge G Barbosa. "Towards high-available and energy-efficient virtual computing environments in the cloud". In: *Future Generation Computer Systems* 40 (2014), pp. 30–43.

[132] Steven. L. Sams. *Discovering hidden costs in your data centre a CFO perspective*. 2011. URL: https://www-935.ibm.com/services/multimedia/data_centre_costs_article_2.pdf (visited on 07/07/2015).

[133] Guntram Scheithauer. "One-Dimensional Bin Packing". In: *Introduction to Cutting and Packing Optimization*. Springer, 2018, pp. 47–72.

[134] Bianca Schroeder, Garth Gibson, et al. "A large-scale study of failures in high-performance computing systems". In: *IEEE Transactions on Dependable and Secure Computing* 7.4 (2010), pp. 337–350.

[135] Mina Sedaghat et al. "DieHard: reliable scheduling to survive correlated failures in cloud data centers". In: *Cluster, Cloud and Grid Computing (CCGrid), 2016 16th IEEE/ACM International Symposium on*. IEEE. 2016, pp. 52–59.

[136] Yogesh Sharma, Amandeep Sharma, and Jyotsna Sengupta. "Performance evaluation of Mobile Ad hoc Network routing protocols under various security attacks". In: *International Conference on Methods and Models in Computer Science (ICM2CS)*. IEEE. New Delhi, India, 2010, pp. 117–124.

[137] Yogesh Sharma et al. “Reliability and energy efficiency in cloud computing systems: Survey and taxonomy”. In: *Journal of Network and Computer Applications* 74 (2016), pp. 66–85.

[138] Yogesh Sharma et al. “Reliable and Energy Efficient Resource Provisioning and Allocation in Cloud Computing”. In: *UCC, 2017 Proceedings ACM*. ACM. 2017, pp. 433–441.

[139] Aidan Shribman and Benoit Hudzia. “Pre-Copy and post-copy VM live migration for memory intensive applications”. In: *Euro-Par 2012: Parallel Processing Workshops*. Springer. 2013, pp. 539–547.

[140] Junaid Shuja et al. “Survey of techniques and architectures for designing energy-efficient data centers”. In: *IEEE Systems Journal* 10.2 (2016), pp. 507–519.

[141] Brian Solomon. *Apple Admits Celebrity Photos Were Stolen In Targeted Hack*. 2014. URL: `https://www.forbes.com/sites/briansolomon/2014/09/02/apple-admits-celebrity-photos-were-stolen-in-targeted-hack/413798973e1a` (visited on 08/11/2015).

[142] Shekhar Srikantaiah, Aman Kansal, and Feng Zhao. “Energy aware consolidation for cloud computing”. In: *Proceedings of the conference on Power aware computing and systems*. ACM. San Diego, California, 2008, pp. 1–10.

[143] Josep Subirats and Jordi Guitart. “Assessing and forecasting energy efficiency on Cloud computing platforms”. In: *Future Generation Computer Systems* 45 (2015), pp. 70–94.

[144] Riky Subrata, Albert Y Zomaya, and Bjorn Landfeldt. “Cooperative power-aware scheduling in grid computing environments”. In: *Journal of Parallel and Distributed Computing* 70.2 (2010), pp. 84–91.

[145] Da-Wei Sun et al. “Modeling a dynamic data replication strategy to increase system availability in cloud computing environments”. In: *Journal of computer science and technology* 27.2 (2012), pp. 256–272.

[146] SK Tesfatsion, Eddie Wadbro, and Johan Tordsson. "A combined frequency scaling and application elasticity approach for energy-efficient cloud computing". In: *Sustainable Computing: Informatics and Systems* 4.4 (2014), pp. 205–214.

[147] Long Thai, Blesson Varghese, and Adam Barker. "A survey and taxonomy of resource optimisation for executing bag-of-task applications on public clouds". In: *Future Generation Computer Systems* 82 (2018), pp. 1–11.

[148] Patrick Thibodeau. *Data centers are the new polluters*. 2014. URL: http://www.computerworld.com/article/2598562/data-center/data-centers-are-the-new-polluters.html (visited on 03/08/2015).

[149] Birjodh Tiwana et al. "Location, location, location!: modeling data proximity in the cloud". In: *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM. Monterey, CA, USA, 2010, pp. 1–6.

[150] Matthew E Tolentino, Joseph Turner, and Kirk W Cameron. "Memory-miser: a performance-constrained runtime system for power-scalable clusters". In: *Proceedings of the 4th international conference on Computing frontiers*. ACM. Ischia, Italy, 2007, pp. 237–246.

[151] Adel Nadjaran Toosi, Rodrigo N Calheiros, and Rajkumar Buyya. "Interconnected cloud computing environments: Challenges, taxonomy, and survey". In: *ACM Computing Surveys (CSUR)* 47.1 (2014), 7:1–7:47.

[152] Kalyanaraman Vaidyanathan et al. "Analysis and implementation of software rejuvenation in cluster systems". In: *ACM SIGMETRICS Performance Evaluation Review* 29.1 (2001), pp. 62–71.

[153] Giorgio Luigi Valentini et al. "An overview of energy efficiency techniques in cluster computing systems". In: *Cluster Computing* 16.1 (2013), pp. 3–15.

[154] Amir Varasteh, Farzad Tashtarian, and Maziar Goudarzi. "On Reliability-Aware Server Consolidation in Cloud Datacenters". In: *arXiv preprint arXiv:1709.00411* (2017).

[155] Blesson Varghese and Rajkumar Buyya. “Next generation cloud computing: New trends and research directions”. In: *Future Generation Computer Systems* 79 (2018), pp. 849–861.

[156] Akshat Verma, Puneet Ahuja, and Anindya Neogi. “pMapper: power and migration cost aware application placement in virtualized systems”. In: *Middleware* 5346 (2008), pp. 243–264.

[157] Kashi Venkatesh Vishwanath and Nachiappan Nagappan. “Characterizing cloud computing hardware reliability”. In: *Proceedings of the 1st ACM symposium on Cloud computing*. ACM. Indianapolis, IN, USA, 2010, pp. 193–204.

[158] Michael Vizard. *$30B Worth of Idle Servers Sit in Data Centers*. 2015. URL: `http://www.datacenterknowledge.com/archives/2015/06/03/report;30b-worth-of-idle-servers-sit-in-data-centers` (visited on 04/30/2018).

[159] William Voorsluys and Rajkumar Buyya. “Reliable provisioning of spot instances for compute-intensive applications”. In: *26th International Conference on Advanced Information Networking and Applications (AINA)*. IEEE. Fukuoka, Japan, 2012, pp. 542–549.

[160] Bimlesh Wadhwa and Amandeep Verma. “Energy and carbon efficient VM placement and migration technique for green cloud datacenters”. In: *Seventh International Conference on Contemporary Computing (IC3)*. IEEE. Noida, India, 2014, pp. 189–193.

[161] John Paul Walters and Vipin Chaudhary. “A fault-tolerant strategy for virtualized HPC clusters”. In: *The Journal of Supercomputing* 50.3 (2009), pp. 209–239.

[162] Chu-Fu Wang, Wen-Yi Hung, and Chen-Shun Yang. “A prediction based energy conserving resources allocation scheme for cloud computing”. In: *International Conference on Granular Computing (GrC)*. IEEE. Noboribetsu, Hokkaido, Japan, 2014, pp. 320–324.

[163] Shun-Sheng Wang and Shu-Ching Wang. “The consensus problem with dual failure nodes in a cloud computing environment”. In: *Information Sciences* 279 (2014), pp. 213–228.

[164] Guoqi Xie et al. “Quantitative Fault-tolerance for Reliable Workflows on Heterogeneous IaaS Clouds”. In: *IEEE Transactions on Cloud Computing* 1 (2017), pp. 1–14.

[165] Lin Yao et al. “Guaranteeing fault-tolerant requirement load balancing scheme based on VM migration”. In: *The Computer Journal* 56.2 (2013), pp. 1–8.

[166] Nezih Yigitbasi et al. “Analysis and modeling of time-correlated failures in large-scale distributed systems”. In: *11th International Conference on Grid Computing (GRID)*. IEEE/ACM. Brussels, Belgium, 2010, pp. 65–72.

[167] John W Young. “A first order approximation to the optimum checkpoint interval”. In: *Communications of the ACM* 17.9 (1974), pp. 530–531.

[168] Xianqing Yu, Peng Ning, and Mladen A Vouk. “Enhancing security of Hadoop in a public cloud”. In: *6th International Conference on Information and Communication Systems (ICICS)*. IEEE. Amman, Jordan, 2015, pp. 38–43.

[169] Longxin Zhang et al. “Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems”. In: *Information Sciences* 379 (2016), pp. 241–256.

[170] Longxin Zhang et al. “Joint optimization of energy efficiency and system reliability for precedence constrained tasks in heterogeneous systems”. In: *International Journal of Electrical Power & Energy Systems* 78 (2016), pp. 499–512.

[171] Longxin Zhang et al. “Maximizing reliability with energy conservation for parallel task scheduling in a heterogeneous cluster”. In: *Information Sciences* 319 (2015), pp. 113–131.

[172] Weiwei Zheng et al. “SPSRG: a prediction approach for correlated failures in distributed computing systems”. In: *Cluster Computing* 19.4 (2016), pp. 1703–1721.

[173] Ao Zhou et al. “Cloud service reliability enhancement via virtual machine placement optimization”. In: *IEEE Transactions on Services Computing* 10.6 (2017), pp. 902–913.

[174] Dakai Zhu, Rami Melhem, and Daniel Mossé. “The effects of energy management on reliability in real-time embedded systems”. In: *IEEE/ACM International Conference on Computer Aided Design (ICCAD-2004)*. IEEE. 2004, pp. 35–40.

[175] Qingbo Zhu and Yuanyuan Zhou. “Power-aware storage cache management”. In: *IEEE Transactions on Computers* 54.5 (2005), pp. 587–602.